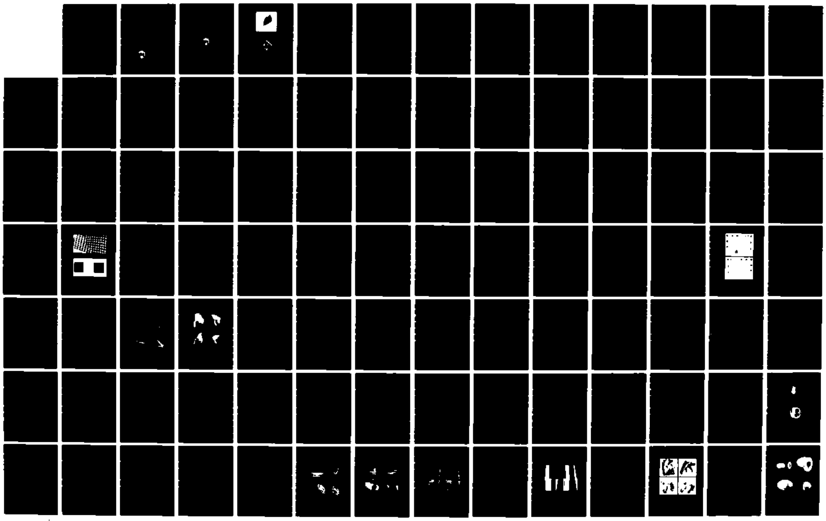
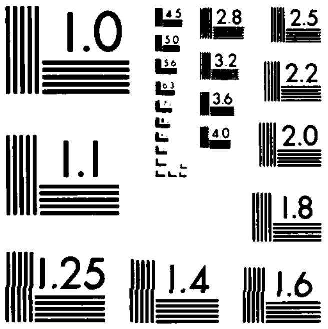
AD-A132 392 GENERATING THREE-DIMENSIONAL SURFACE MODELS OF SOLID OBJECTS FROM MULTIPL..(U) RENSSELAER POLYTECHNIC INST TROY NY IMAGE PROCESSING LAB M POTMESIL OCT 82 1/3

UNCLASSIFIED IPL-TR-033 N00014-82-K-0301 F/G 9/2 NL





1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

12

IPL-TR-033

GENERATING THREE-DIMENSIONAL SURFACE MODELS OF SOLID OBJECTS FROM MULTIPLE PROJECTIONS

Michael Potmesil

October 1982

Image Processing Laboratory

Rensselaer Polytechnic Institute
Troy, New York 12181

IPL-TR-033

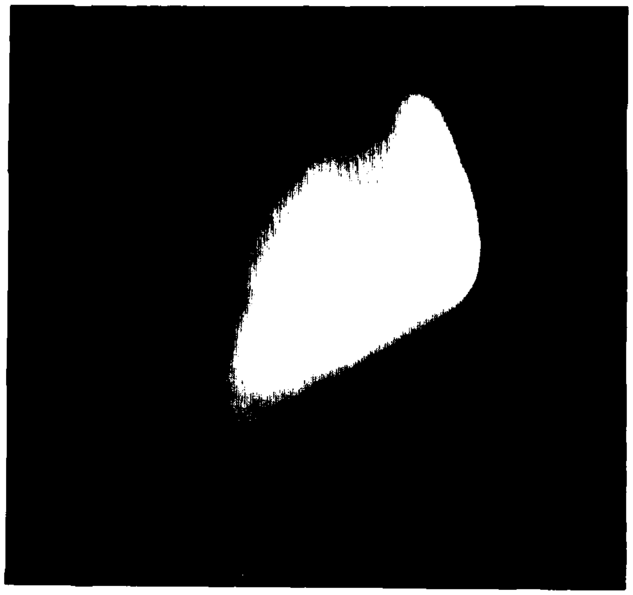
GENERATING THREE-DIMENSIONAL SURFACE MODELS OF SOLID OBJECTS FROM MULTIPLE PROJECTIONS

Michael Potmesil

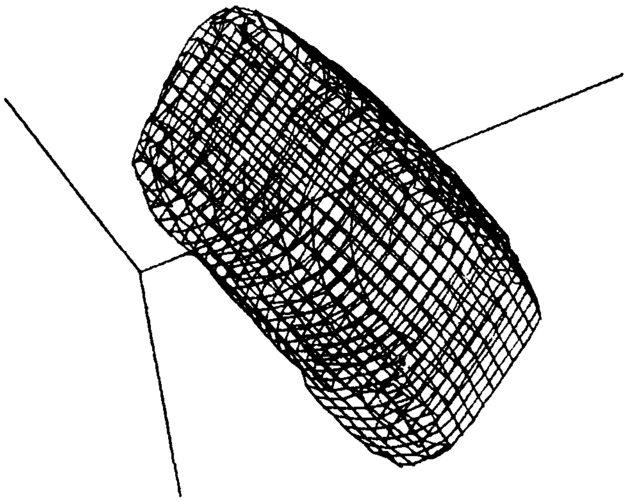
October 1982

IMAGE PROCESSING LABORATORY

Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12181



(a)



(b)

Computer-generated model of a three-dimensional object obtained from 36 digital images: (a) a view of the actual object, (b) a view of the model

ABSTRACT

A recurring problem in computer vision and related fields is that of generating computer models of physical objects. This thesis presents a method for constructing such models in the form of three-dimensional surface or volume descriptions. The surface models are composed of curved, topologically rectangular, parametric patches. The data required to define these patches are obtained from multiple photographic images of the object illuminated by a rectangular pattern of lines. The projection of the pattern on the surface of the object traces curves which define the boundaries of the patches. The 3D description of the patches is reconstructed by photogrammetric techniques from two or more images of the projected pattern. A calibration stand, in which the object is placed, permits determination of the camera geometry directly from image data.

This method generates 3D surface descriptions of only those parts of the object that are illuminated by the projected pattern, and also are visible in at least two images. A complete model of the object is obtained by repeating this reconstruction process for various arbitrary orientations of the object until descriptions covering the entire surface have been obtained. Since each description is defined in its own 3D object space and 2D parameter space, a surface-matching procedure is developed to register spatially the surface descriptions in a common object space. This procedure searches for a 3D rigid transformation of the surface descriptions which minimizes their shape difference. Once the surface descriptions are in the same object space, they are also merged into a common parameter space. This match-and-merge process is iteratively repeated for pairs of surface descriptions until a complete model of the object is assembled.

ACKNOWLEDGMENT

The author wishes to thank Professor Herbert Freeman for his guidance, encouragement, support, and patience during the long course of this work. Many endless discussions with Indranil Chakravarty on topics related to this work are acknowledged. He also co-authored the ever-growing series of papers on computer-made fuzzy images. Thanks are also due to Al Barr for providing mathematical assistance, to Ernesto Guerrieri for extending help on many occasions, and to Don Meagher for growing the octrees.

Joachim Frank and Adriana Verschoor of the New York State Department of Health kindly provided the equipment and assistance in digitizing the photographic images. Del Coates of College of Art and Design designed the car model used as an example in this work. Joe Mundy of General Electric Corporate Research and Development Center provided the blade models.

The research described here was supported in part by the National Science Foundation, Automation, Bioengineering and Sensing Program, under Grant ENG-7904821, and in part by the Office of Naval Research under Contract N00014-82-K-0301. This support is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

Figure Page

LIST OF SYMBOLS

$i=1,2,..,I$	Index of image coordinate system i
$j=1,2,..,J$	Index of parameter coordinate system j
$k=1,2,..,K$	Index of object coordinate system k
$Q(r,s,w)_i$	Homogeneous 2D coordinate system defining image space i
$P(u,v,w)_j$	Homogeneous 2D coordinate system defining parameter space j
$O(x,y,z,w)_k$	Homogeneous 3D coordinate system defining object space k
$T\{P_j \rightarrow O_k\}$	Transformation from parameter space P_j to object space O_k
$T\{O_k \rightarrow Q_i\}$	Transformation from object space O_k to image space Q_i
$T\{O_k \rightarrow O_{k'}\}$	Transformation from object space O_k to object space $O_{k'}$
$T\{P_j \rightarrow P_{j'}\}$	Transformation from parameter space P_j to parameter space $P_{j'}$
$f(x,y,z,w)$	Algebraic representation of a surface element
$\overline{A}$	Matrix of algebraic surface representation coefficients
$\overline{g}(u,v)$	Parametric representation of a surface element
$\overline{B}$	Matrix of parametric surface representation coefficients
$\overline{h}(x,y,z,w)$ $\overline{h}(u,v,w)$ $\overline{h}(r,s,w)$	A point in object, parameter, and image coordinate systems, respectively
$\overline{c}(w)$	Parametric representation of a curve (with w as the parameter)
$\overline{C}$	Matrix of parametric curve representation co-

	efficients
$\overline{l}(w)$	Parametric representation of a line (with w as the parameter)
$\overline{L}$	Matrix of parametric line representation coefficients
$m=0,1,\ldots,M$	Index of patch control points and curves along parameter u
$n=0,1,\ldots,N$	Index of patch control points and curves along parameter v
G	Graph of surface representation
$\overline{N}$	Surface normal vector
$\overline{U}$, $\overline{V}$, $\overline{W}$	Cubic basis vectors [u^3 u^2 u 1], [v^3 v^2 v 1], and [w^3 w^2 w 1], respectively
e(....)	Evaluation function
$\|a\|$	Absolute value of a
$\overline{a} \cdot \overline{b}$	Scalar dot product of vectors $\overline{a}$ and $\overline{b}$
$\overline{a} \times \overline{b}$	Vector cross product of vectors $\overline{a}$ and $\overline{b}$
a | b	Logical OR of a and b
a & b	Logical AND of a and b

CHAPTER 1

INTRODUCTION

1.1 Statement of the Problem

A recurring problem in computer vision and related fields has been automatic generation of computer models of the surfaces of arbitrarily-shaped, physical, three-dimensional objects. Generation of such models inherently requires the acquisition and analysis of 3D surface data. In this context, acquisition refers to the ability to enter automatically numerical data describing 3D surfaces into a computer data base, and analysis refers to the ability to organize this data base so that it contains concise and efficient models of the complete surfaces of 3D objects. The models in such a data base are then available to various application tasks. The objective of the work described here was to develop computer techniques for the construction and processing of such 3D surface models.

A model of a 3D object, such as of a machine part, can be used in a variety of computer tasks. In computer-aided design (CAD), a clay model of a new part or an existing part is entered into a computer data base for engineering studies such as finite-element analysis (FEA). Further modifications to the shape of the part are made only to the computer model. In computer-aided manufacturing (CAM), the computer model is used to manufacture the part by a numerically-controlled machine. In robotics, a mechanical manipulator with visual or tactile capabilities recognizes, inspects and assembles the part using the model. In computer graphics, the model is used for animated display of 3D simulation of these tasks.

In general, the surfaces being considered here may vary from those of man-made objects such as industrial parts or broken pottery pieces to those of natural objects such as earth terrain, and the techniques developed here are applicable to fields such as automation, archeological restorations, terrain recognition, and vision for intelligent robots. This work also attempts to bridge the gap between the approaches to computer modeling of 3D objects in CAD/CAM (object synthesis), and those in computer vision (object analysis).

1.2 Outline of the Approach

The process of constructing computer models of surfaces of three-dimensional objects, developed in this work, is divided into three major parts:

(1) generation of a numerical description of the shape of a surface;

(2) hierarchical structuring of this surface descrip-

tion to allow efficient processing; and

(3) matching and merging of partial surface descriptions into a complete surface model of an object.

The surface representation, selected here, uses the _bicubic parametric patch_ as the basic _surface element_. Adjacent patches with positional and derivative continuities across their boundaries form _sheets of composite patches_. Given enough of these composite patches, an arbitrarily-shaped surface can be represented to an arbitrary precision. If necessary, however, this representation can be expanded to higher order bivariate polynomials, or simplified to bilinear or planar patches.

The 3D surface data, required to define composite bicubic patches, are obtained by photogrammetric methods from multiple digital images of surfaces illuminated by a two-dimensional parameter space. The parameter space contains a pattern of orthogonal lines, defined on a unit-square grid. The projected lines of the parameter space trace 3D parametric curves on the measured surface. These curves are the boundary curves of the surface patches being computed. The intersection points of these curves are the control points of the surface patches. By identifying the two-dimensional projections of these points and curves in two or more images, they can be reconstructed into the original 3D object space. The reconstructed 3D control points or boundary curves are then converted into linear lists of 3D bicubic parametric patches.

Having generated a 3D surface description made of composite patches, a hierarchical control structure that allows an efficient representation and processing of these patches is developed. Since the patches are parametrized in a two-dimensional parameter space where the domain of each patch is a unit square, a hierarchical structure in the form of a surface quadtree is utilized. A quadtree of composite patches is built by recursively merging the four patches that share a common control point into a new patch at the next higher level of the quadtree. Each node in the quadtree consists of a description of its patch, a bounding volume which encloses the patch, and the average normal vector of the normal vector field of the patches below the current quadtree node. The bottom level of the quadtree contains the original patches that were generated from the parametrized surfaces. The higher levels contain coarser and coarser approximations to the parametrized surfaces. This quadtree format of bivariate surface elements allows, in general, logarithmic rather than linear searching and sorting time of the individual elements.

The modeling method described up to now generates only a description of a surface segment that is parametrized by a projected pattern of lines, and also visible in at least two images of the surface. Several such segments must be obtained to cover the complete surface of a typical 3D solid object. They are obtained by changing the object's orientation and repeating the surface acquisition process. Each surface segment is, therefore, defined in its own 3D object space. The surface segments are obtained, however, in such a way that they partially overlap to allow their eventual alignment by matching the common surface sections.

As the last and, actually, the main step in this modeling process, we match the individual surface segments of an object in a common 3D space, and then also merge them into a common 2D parametric space. A search procedure was developed to register spatially two partially overlapping surface segments at a time. It employes a heuristic-search algorithm with an evaluation function to compute a rigid 3D transformation, which minimizes a set of shape-feature distances between the two surfaces. The shape-feature distances can be evaluated at (1) patch control points, (2) points with maximum surface curvature, or (3) points identified and matched in images of the surfaces. These feature points are selected at each level of a surface quadtree. The search algorithm computes distances from the feature points on one surface to the other surface by tracing a ray normal to the first surface at a feature point and intersecting it with the other surface. From the distance, angle of intersection, and the curvature of the second surface at the point of intersection, the algorithm evaluates the surface match of the current node in the search tree and generates new 3D transformations for the successor nodes in the search process. The 3D transformations of the initial nodes of the search tree are generated by aligning the normal vectors in the top nodes of the two quadtrees being matched.

After the two surface segments have been transformed into the same object space, they must be merged into a common parameter space. Several algorithms, which merge overlapping surface segments using either transformation of parameters or projection of parameters, have been developed. This match-and-merge process is iteratively repeated for all the 3D surface segments until a complete model of the object, in a single 3D object space and a single 2D parameter space, is generated. A surface model of a solid object can then be converted into a volume model.

The surface-matching procedure is also generalized to perform these two tasks:

(1) <u>surface and object recognition</u> - a partial 3D surface description or a complete object model is matched against a set of complete object models to determine whether the surface may possibly be a part of one of the objects; and

(2) <u>surface and object segmentation into simple surface and volume shapes</u> - a 3D surface description or a 3D surface model of a complete object is matched with surface or volume shape primitives; once a

match is obtained, the matched region is segmented and the process may again be iteratively repeated in order to decompose a complicated object into a structure of simple shapes.

The following work, although not directly a part of the modeling process, has also been performed, and is briefly presented in this report:

(1) generation of synthetic raster images to allow a realistic display of the modeled surfaces;

(2) expansion of the traditional pin-hole camera model for display of synthetic raster images to a model of camera's optical system which includes the notion of focusing, depth of field, and motion blur caused by a lens, an aperture opening, and a finite exposure time, respectively; and

(3) development of the software systems which implement the modeling process as well as the synthetic image generation and the optical camera model.

Scene analysis, image processing and computer graphics are mostly applied sciences. They show that something is possible by doing it rather than by formally proving that it can be done. The work described here is presented in this spirit by ommiting formal assumptions, proofs, lemmas or anything else that could just make it look better than it actually is. It contains descriptions of techniques, algorithms and data structures and reports on how well they do or do not work.

This report has the following structure: the three steps of the modeling process are presented in the next three chapters; each of these chapters is illustrated with several examples; then a chapter is devoted to the hardware employed and the software developed for the modeling process. Finally, a chapter with comprehensive results of the modeling process follows.

1.3 Literature Survey

There is an extensive amount of recent literature that is pertinent to this project in the areas of image processing, scene analysis, robotics, artificial intelligence and computer graphics. The following brief survey is divided into three sections which correspond to the three major steps (surface acquisition, representation, and matching) in building object models as described in this work.

1.3.1 Surface Acquisition

There are a number of techniques for automatic, non-contact surface digitization based on (1) light reflection from controlled light sources, (2) time-of-flight measurements, and (3) stereometric correlations. Illumination of a

3D object by a line pattern was first reported in the image processing literature by Will and Pennington [73,74], who used it to locate planar faces of polyhedral objects. A normal vector to a planar face was computed from a 2D Fourier transform of a single image illuminated by a grating of parallel lines. Idesawa et al [38,39] computed 3D surface shape from Moire fringe patterns. These patterns are formed when an object illuminated by a grating is observed through another grating. Scene analysis using 3D data obtained with a range finder can be found in several papers by Agin [1,2], Oshima and Shirai [51], and Tsukiama [71]. Typically, a range finder computes 3D depth information from a single line of light projected on a scene. Horn [35] obtained surface geometry from single images by analyzing illumination, reflectance and surface shading of a scene. Posdamer and Altschuler [53] proposed a laser shutter/space encoding system capable of digitizing several thousand 3D surface points in real time.

Reconstruction of objects and surfaces from multiple views has been performed by several authors. Rabinowitz [61] reconstructed vertices and edges of polyhedral objects, Shapira [68] matched regions of objects bounded by quadric surfaces, Baker [6] constructed models of objects from multiple 2D silhouettes. England [25] interactively fitted 3D surface description on two stereo views of an object. Burr [13] matched two stereo images. Fuchs et al [30] reconstructed objects from series of planar contours. A survey by Bajcsy [5] lists other current scene-analysis methods for acquisition of 3D data using monocular and binocular depth cues.

1.3.2 Surface Representation

Parametric curved-surface representation was developed by Coons [20]. The bicubic parametric patch has been extensively used in computer-aided design [7,16,26] and in computer graphics [15,40]. Other surface representations using planar and quadric surfaces were studied by Baumgard [10] and Levin [41], respectively. Barr [8] extended the quadric representations into a family of more powerful shapes. Volumetric representation of objects was used by Agin and Binford [2] (generalized cylinders), O'Rourke and Badler [50] (spheres), and Meagher [44] (octal trees of cubes). Hierarchical surface representation using bounding volumes was proposed by Clark [18] to improve hidden-surface algorithms for complex 3D scenes, and implemented by Whitted [72] and Rubin and Whitted [64]. Newell [47], Grossman [33], and Marshall et al [45] utilize procedural representations of 3D objects. Synthetic object modeling for programmed assembly by mechanical manipulators can be found in works by Grossmann and Taylor [32], and Lozano-Perez and Winston [42]. A survey of existing systems for geometric modeling in computer-aided design is presented by Baer et al [4]. Badler and Bajcsy [3] survey methods of representing 3D objects for computer graphics and computer vision.

1.3.3 Surface Matching

Most of the work in surface matching and shape analysis has been primarily done by comparison of 2D shapes or 3D shapes projected into 2D images. For example, Davis [21] used relaxation labeling of local features for 2D shape matching. Burr [14] proposed a dynamic elastic model for image as well as line-drawing matching.

Relevant research in 3D matching was done by Baker [6] who compared 3D shapes made of piece-wise surface primitives. Barrow et al [9] presented a technique of matching a 3D description of a scene and its 2D images. Horn and Bachman [36] aligned a synthetically shaded image of a 3D terrain model with an actual image of the terrain. Burr [13] matched reconstructed 3D edges of objects with stored wire frame models utilizing 3D features and geometrical constraints. Shapiro et al [69] matched structured descriptions of objects.

1.4 Summary of Definitions

The modeling process described here utilizes three coordinate systems and four mappings between these systems. We need to define these systems and mappings in detail before proceeding further:

$O(x,y,z,w)_k$ is a homogeneous <u>3D object coordinate system</u> which defines object space <u>k</u>. All 3D representations of surface elements are specified in an object coordinate system. An object being modeled is positioned in object coordinate spaces <u>k</u> = 1, 2, ..., <u>K</u>. Notation $O(x,y,z,w)_{j,k}$ means that object space <u>k</u> is parametrized by parameter space <u>j</u>.

$P(u,v,w)_j$ is a homogeneous <u>2D parameter coordinate system</u> which defines parameter space <u>j</u>. A set of orthogonal lines on a unit-square grid parametrizes a 3D surface in an object space. An object being modeled is parametrized by parameter coordinate systems <u>j</u> = 1,2, ..., <u>J</u>. There is always at least <u>one</u> parameter coordinate system <u>j</u> for every object coordinate system <u>k</u>. Notation $P(u,v,w)_{j,k}$ means that parameter space <u>j</u> parametrizes object space <u>k</u>.

$Q(r,s,w)_i$ is a homogeneous <u>2D image coordinate system</u> which defines image space <u>i</u>. An object being modeled is observed in image coordinate systems <u>i</u> = 1, 2, ..., <u>I</u>. There are always at least <u>two</u> image coordinate systems <u>i</u> and <u>i</u>' for every object coordinate system <u>k</u> and parameter coordinate system <u>j</u>. Notation $Q(r,s,w)_{i,j,k}$ means that object space <u>k</u> is parametrized by parameter space <u>j</u> and observed in image space <u>i</u>.

$T\{P_j \rightarrow O_k\}$ is a mapping from a 2D parameter space P_j to a 3D object space O_k. This mapping takes places (via a projector) when 2D lines of a parameter space are projected on a 3D object space.

$T\{O_k \rightarrow Q_i\}$ is a mapping from a 3D object space O_k to a 2D image space Q_i. This mapping takes place (via a camera) when a 2D projection of a parametrized surface is recorded in a 2D image coordinate system.

$T\{O_k \rightarrow O_{k'}\}$ is a mapping from object space O_k to object space $O_{k'}$. This mapping is computed by the matching procedure to transform two 3D surface descriptions into a common object space.

$T\{P_j \rightarrow P_{j'}\}$ is a mapping from parameter space P_j to parameter space $P_{j'}$. This mapping is computed by the merge procedure to transform a 3D surface, parametrized in two different parameter spaces, into a common parameter space.

Additionally, the following terms are used throughout this report:

A _parametrizing grid_ is a set of orthogonal 2D isoparametric lines used to parametrize a 3D surface.

An _isoparametric line network_ is a 2D image projection of a parametrizing grid projected on a 3D surface.

A _surface element_ is a 3D surface representation by a planar face, a sphere, a quadric surface, or a bicubic patch.

A _bounding surface_ is a surface element used solely as a bound of other surface element or elements.

A bounding volume is a set of one or more surface elements which completely enclose a 3D space. One or more surface elements or other bounding volumes are completely circumscribed by a bounding volume. Typical bounding volumes are a sphere, an ellipsoid or a parallelepiped.

A surface quadtree is a hierarchical surface structure with surface representation parametrized by two orthogonal parameters which are defined in a 2D plane. A node of the tree has up to four successors which contain descriptions of the current surface element subdivided into four more detailed elements.

A sheet of composite patches is a set of contiguous patches with adjacent patches having at least first-derivative continuity (C^1) across their boundaries. All patches in a sheet are parametrized by the same parameter coordinate system. A sheet of patches is stored in a single surface quadtree. Patches in different sheets of an object have at most positional continuity (C^0).

A surface segment is the part of a surface that is described in one object coordinate system and one parameter coordinate system.

A surface model of a solid object is a union of sheets of composite patches which define the surface of the object.

A volume model of a solid object is a union of rectangular parallelepipeds which define the volume of the object.

CHAPTER 2

GENERATION OF 3D SURFACE DESCRIPTIONS

This chapter describes a method for making automatic, non-contact measurements of the surfaces of physical, rigid, arbitrarily-shaped 3D objects [54,28,56]. Surface information is obtained by photogrammetric and image-processing techniques from multiple images of the measured surfaces illuminated by a controlled light source. The method, as used here, generates surface data that are used in modeling the surfaces with composite bicubic patches. However, it can also be adapted to surface modeling with surface elements ranging from planar triangles to high-order polynomials.

Surface representation by the bicubic parametric patch, as defined in detail in Appendix A, has been chosen because of its widespread applications in the fields where computer-generated models of 3D objects are extensively used, such as CAD/CAM and computer graphics. The format of the patch employed here can be defined either by a set of 3D control points or by a set of 3D boundary curves. The photogrammetric reconstruction method allows us to compute either the control points or the boundary curves from multiple images of the surface-patch area.

The first three sections of this chapter contain the background mathematics necessary for the reconstruction process. The fourth section then describes the actual computer methods implementing this process.

2.1 2D to 3D Parameter Mapping

To model an existing 3D surface with parametric patches, the surface is parametrized with orthogonal isoparametric lines. $T\{P_j \rightarrow O_k\}$ maps a 2D parameter homogeneous coordinate system $P(u,v,w)_j$ into a 3D object homogeneous coordinate system $O(x,y,z,w)_k$ [Figure 2.1]:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}_k = T\{P_j \rightarrow O_k\} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_j = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \\ t_{41} & t_{42} & t_{43} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_j \qquad (2.1)$$

The parametric information mapped on the surfaces in the object space can be described by a set of orthogonal lines and their intersection points. There are ($\underline{M}$+1) x ($\underline{N}$+1) isoparametric lines; those with $\underline{u}$ as the parameter and $\underline{v}$ constant are labeled:

$\bar{I}_0(u), \ldots, \bar{I}_n(u), \ldots, \bar{I}_N(u)$,

and those with $\underline{v}$ as the parameter and $\underline{u}$ constant are labeled:

$\bar{I}_0(v), \ldots, \bar{I}_m(v), \ldots, \bar{I}_M(v)$.

The intersection point of two orthogonal lines $\bar{I}_n(u)$ and

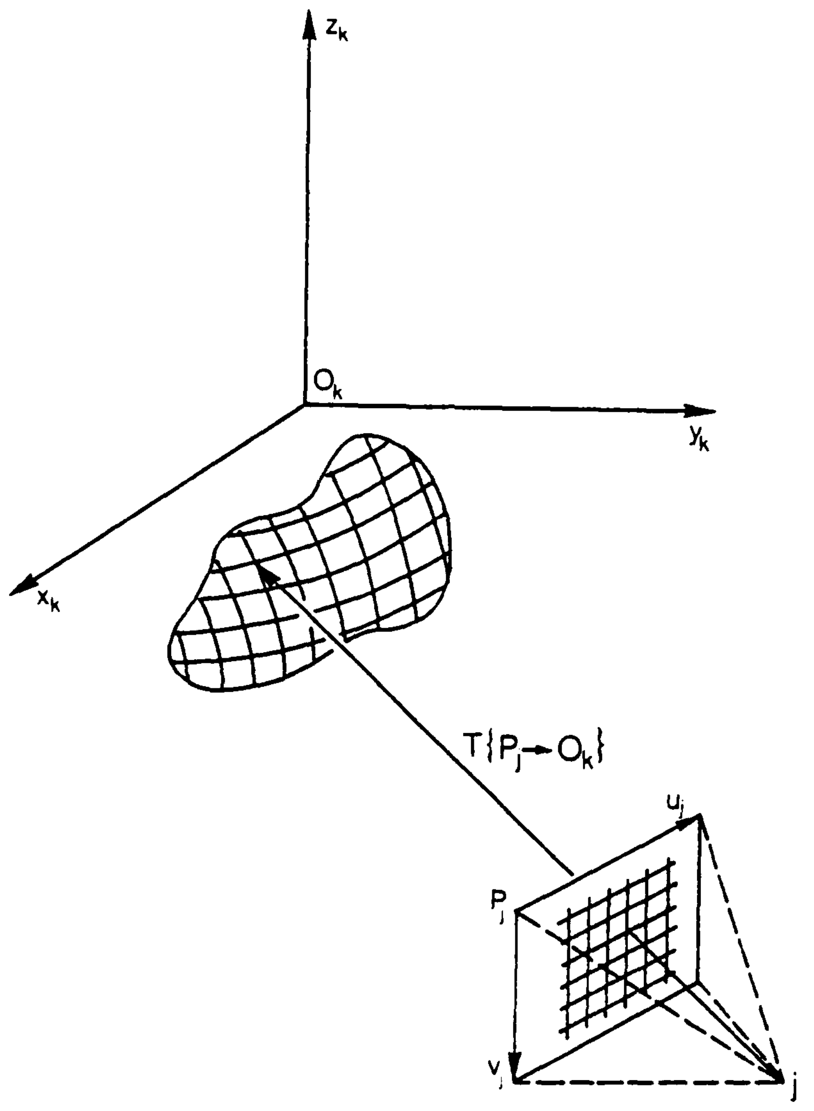


Figure 2.1 Object and parameter coordinate systems and mapping $T\{P_j \rightarrow O_k\}$

$\overline{l}_m(v)$ is a patch control point $\overline{h}_{m,n}(u,v,w)$ [Figure A.1(b)]. This point is mapped on surface point $\overline{h}_{m,n}(x,y,z,w)_k$ and then projected into image point $\overline{h}_{m,n}(r,s,w)_i$. It becomes a control point of patch $\overline{g}_{m,n}(u,v)$. The line segment from point $\overline{h}_{m,n}(u,v,w)$ to point $\overline{h}_{m,n+1}(u,v,w)$ is $\overline{l}_{m,n}(v)$. It is mapped into a 3D patch boundary curve $\overline{c}_{m,n}(v)_k = [\ x(v)\ y(v)\ z(v)\ w(v)\ ]_k$ and then projected into a 2D image curve $\overline{c}_{m,n}(v)_i = [\ r(v)\ s(v)\ w(v)\ ]_i$. The other line segment from point $\overline{h}_{m,n}(u,v,w)$ to point $\overline{h}_{m+1,n}(u,v,w)$ is similarly labeled. The mappings of these two line segments become boundary curves of patch $\overline{g}_{m,n}(u,v)$. This defines the surface information required to construct an array (a sheet) of $\underline{M}$ x $\underline{N}$ parametric patches.

2.2 3D to 2D Image Mapping

Transformation $T\{O_k \rightarrow Q_i\}$ maps a 3D object homogeneous coordinate system $O(x,y,z,w)_k$ to a 2D image homogeneous coordinate system $Q(r,s,w)_i$ [Figure 2.2]:

$$\begin{bmatrix} rw \\ sw \\ w \end{bmatrix}_i = T\{O_k \rightarrow Q_i\} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_k = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_k \tag{2.2}$$

The transformation $T\{O_k \rightarrow Q_i\}$ consists of translations

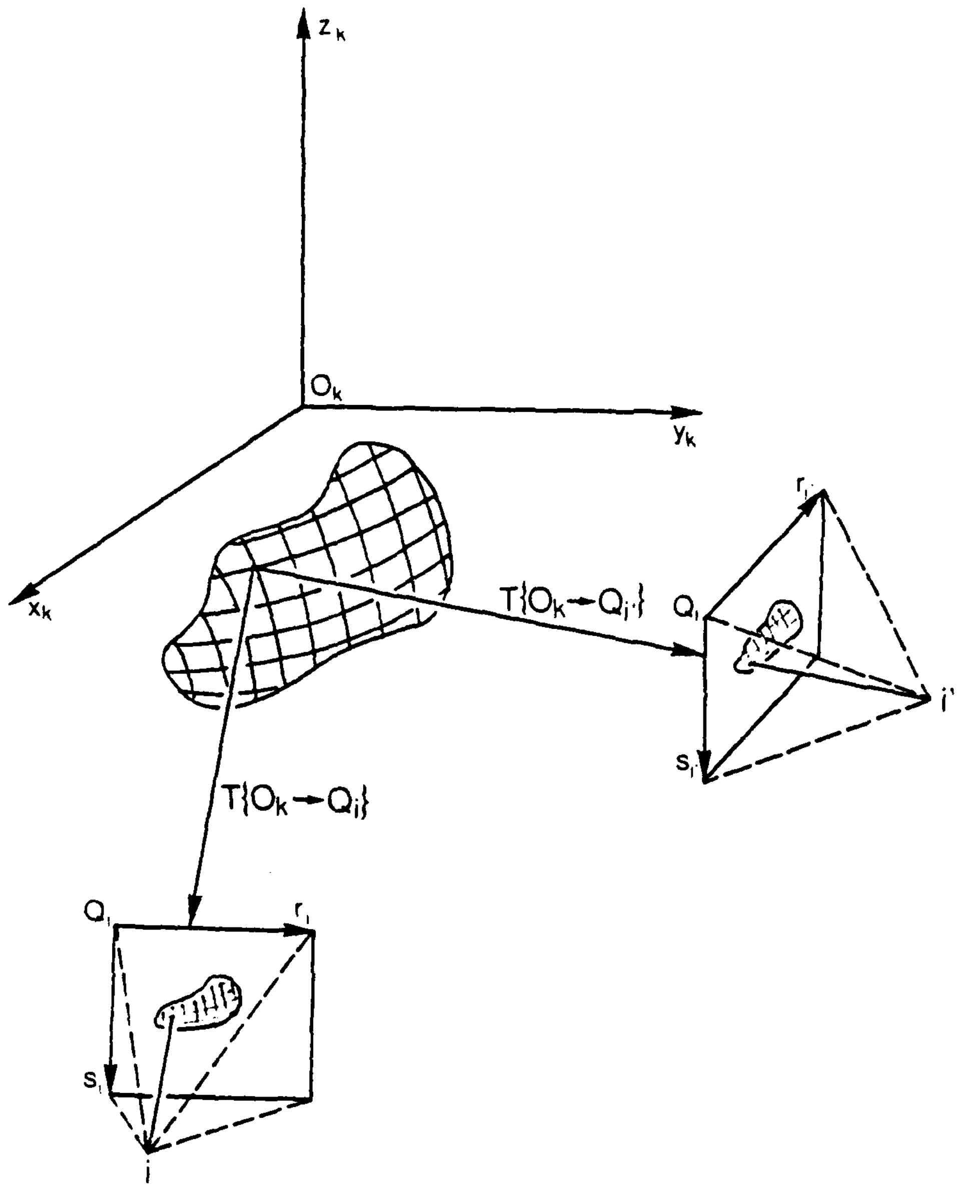


Figure 2.2 Object and image coordinate systems and mappings $T\{O_k \rightarrow Q_i\}$ and $T\{O_k \rightarrow Q_{i'}\}$

and rotations of the camera in $O(x,y,z,w)_k$, perspective projection, and conversion into the digitized image coordinate system $Q(r,s,w)_i$. Its detailed composition has been previously discussed in [54].

2.3 2D to 3D Surface Reconstruction

To map surface information from an image plane $Q(r,s,w)_i$ back into the 3D space $O(x,y,z,w)_k$ we have to find an inverse mapping to that given in equation (2.2). For the 2D projection $\overline{h}(r,s,w)_i$ in $Q(r,s,w)_i$ of a 3D point $\overline{h}(x,y,z,w)_k$ in $O(x,y,z,w)_k$ and mapping $T\{O_k \rightarrow Q_i\}$ from the 3D object coordinate system to the 2D image coordinate system, we can rewrite equation (2.2) into a system of two linear equations with three unknowns:

$$\begin{bmatrix} r_i t_{31}-t_{11} & r_i t_{32}-t_{12} & r_i t_{33}-t_{13} & r_i t_{34}-t_{14} \\ s_i t_{31}-t_{21} & s_i t_{32}-t_{22} & s_i t_{33}-t_{23} & s_i t_{34}-t_{24} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_k = \overline{0} \tag{2.3}$$

The solution to this system of equations produces only the equation of the 3D line $\overline{h}(r,s,w)_i$ $\overline{h}(x,y,z,w)_k$. To compute the coordinates of the point $\overline{h}(x,y,z,w)_k$ we need an additional projection $\overline{h}(r,s,w)_{i'}$ in $Q(r,s,w)_{i'}$ and mapping $T\{O_k \rightarrow Q_{i'}\}$ as shown in Figure 2.2. With two projections we obtain four equations with three unknowns and the coordinates of point $\bar{h}(x,y,z,1)_k$ can be solved. The least-square error solution of the system in equation (2.3) for $\underline{I}$ projections was previously described in [54]. The reconstruction of surface points, presented in this section, can be generalized to reconstruction of surface curves. Appendix B contains an analogous method for the reconstruction of a 3D parametric cubic curve from its multiple projections in 2D images.

The reconstruction error of a point $\bar{h}(x,y,z,1)_k$ is evaluated from equation (2.2) by projecting the reconstructed point back into each image $\underline{i}$ from which it was obtained, and computing the 2D image distance between this projection and the originally measured point $\bar{h}(r_i,s_i,1)_i$. The total error is then the sum of errors in the $\underline{I}$ images:

$$d=\sum_{i=1}^{I}\sqrt{\left(r_i - \frac{t_{11}x+t_{12}y+t_{13}z+t_{14}}{t_{31}x+t_{32}y+t_{33}z+t_{34}}\right)^2 + \left(s_i - \frac{t_{21}x+t_{22}y+t_{23}z+t_{24}}{t_{31}x+t_{32}y+t_{33}z+t_{34}}\right)^2} \tag{2.4}$$

In the above equation (2.4) the coefficients $\underline{t}$ belong to the camera transformation matrix $T\{O_k \rightarrow Q_i\}$. Note also that the error $\underline{d}$ is given in pixel units of the image coordinate systems.

The twelve coefficients of the camera transformation matrix $T\{O_k \rightarrow Q_i\}$ in equation (2.3) are computed for each age $\underline{i}$ by a camera calibration method [70]. Equation (2.2) may be rewritten as two simultaneous linear equations by eliminating $\underline{w}$:

$$\begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & -xr & -yr & -zr & -r \\ 0 & 0 & 0 & 0 & x & y & z & 1 & -xs & -ys & -zs & -s \end{bmatrix} \begin{bmatrix} t_{11} \\ t_{12} \\ \cdots \\ \cdots \\ t_{33} \\ t_{34} \end{bmatrix} = \overline{0} \tag{2.5}$$

where $\underline{x}$, $\underline{y}$, $\underline{z}$, are the coordinates of a point $\overline{h}(x,y,z,1)_k$ in $O(x,y,z,w)_k$ and $\underline{r}$, $\underline{s}$ are the coordinates of its image $\overline{h}(r,s,1)_i$ in $Q(r,s,w)_i$. There are twelve unknown $\underline{t}$'s in the two equations of (2.5). If we know six points in $O(x,y,z,w)_k$ and their corresponding images in $Q(r,s,w)_i$, we can create twelve simultaneous equations (two for each point) and solve for the unknown coefficients. In order to solve such a system of linear equations, it has to be made non-homogeneous be letting, for example, $t_{34} = 1$. This is allowed since the homogeneous coordinate systems that we are using have an arbitrary scale that is set to 1.

In general, for six or more calibration points the least-squares error method is again used to solve a system of eleven non-homogeneous equations (with $t_{34} = 1$) [54]. Once the coefficients of $T\{O_k \rightarrow Q_i\}$ are computed, the error of each calibration point may be checked by evaluating equation (2.2) for $\overline{h}(x,y,z,1)_k$ and comparing the computed point $\overline{h}(r,s,1)_i$ with the measured calibration point $\overline{h}(r,s,1)_i$. Calibration points with large errors may then be disregarded and the procedure repeated for a better estimate of $T\{O_k \rightarrow Q_i\}$.

2.4 Photogrammetric Reconstruction

In the approach, taken here, the surface to be digitized is illuminated by the parametrizing pattern described in Section 2.1. The pattern is focused on the 3D surface and its shadow traces 3D isoparametric curves there. Using image processing and photogrammetric techniques, a 3D description of the pattern projected on the surface can be obtained. Therefore, the 3D surface being measured is illuminated by a 2D pattern of parametric lines $P(u,v,w)_j$ with a projector located at $\underline{j}$ [Figure 2.1]. The illuminated parts of the surface are photographed from at least two locations $\underline{i}$ and $\underline{i}'$ by a camera onto 2D image planes $Q(r,s,w)_i$ and $Q(r,s,w)_{i'}$, respectively [Figure 2.2]. The 3D object coordinate system $O(x,y,z,w)_k$ is defined by a set of camera calibration marks placed in a camera calibration stand. The measured surface is within this stand.

This arrangement limits us to surfaces that can be photographed under the conditions imposed by this method. That is, the surfaces are photographed in a camera calibration stand; they can not be transparent, have high specular reflection, or be totally black. This method also requires a large amount of computations, and image digitization at a high resolution to obtain precise surface measurements. It is mainly intended for surface measurements of objects whose models will later be used in various applications, rather than for real-time surface acquisition and processing.

The following is an outline (specified in a home-grown variation of the "Algol-like notation") of the reconstruction algorithms, once all $\underline{I}_{j,k}$ digitized images for a single parameter coordinate system $\underline{j}$ and a single object coordinate system $\underline{k}$ have been obtained:

(2.6)

```
procedure extract($I_{j,k}$);
   begin
   for i := 1 step 1 until $I_{j,k}$ do
      begin
      extract coordinates of camera calibration marks;
      match them with their coordinates in $O(x,y,z,w)_k$;
      compute $T\{O_k \rightarrow Q_i\}$;
      extract the projected line pattern;
      end;
   return;
   end;
```

and then

(2.7)

```
procedure reconstruct($I_{j,k}$);
   begin
   match extracted patterns for images 1 to $I_{j,k}$;
   reconstruct matched surface data for images 1 to $I_{j,k}$;
   return;
   end;
```

Images of surfaces are taken by a camera on monochrome film and digitized. There are two types of data that need to be extracted from a digitized image: the projected line patterns and the illuminated calibration marks. Both data types can be stored in binary images (each pixel is either 0 or 1), where the projected pattern is a network of black lines on a white surface with black background [Figure 2.3(a)], and the calibration marks are white marks on black background [Figure 2.3(b)].

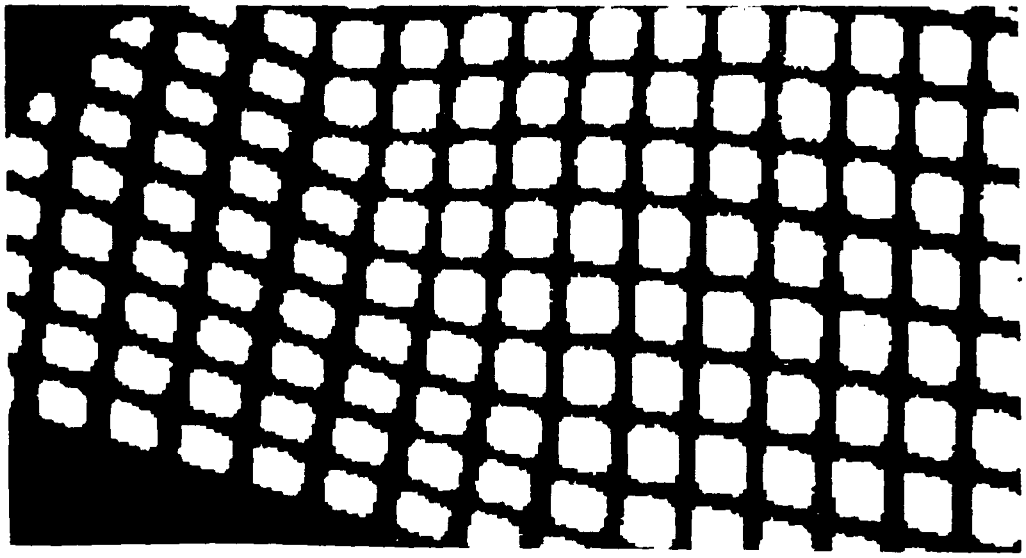
The projector illuminating the measured surface can be considered a single point-light source. The light reflection from the surface is only diffuse reflection; any potential specular reflection can be supressed by the use of polarizing filters. Therefore the light intensity in an image can be approximately modeled by Lambert's law:

$$\text{intensity} = k_d \, (\overline{N} \cdot \overline{H}) \tag{2.8}$$

where

k_d	=	diffuse reflection coefficient
$\overline{N}$	=	surface normal vector
$\overline{H}$	=	light source vector
intensity	=	final image intensity

It is apparent from equation (2.8) that image intensity and, therefore, also the contrast of the projected lines decrease as the illuminated surface turns away from the projector. Simultaneously, the spacing of the lines projected into the image decreases as does their width. A frequency-


(a)

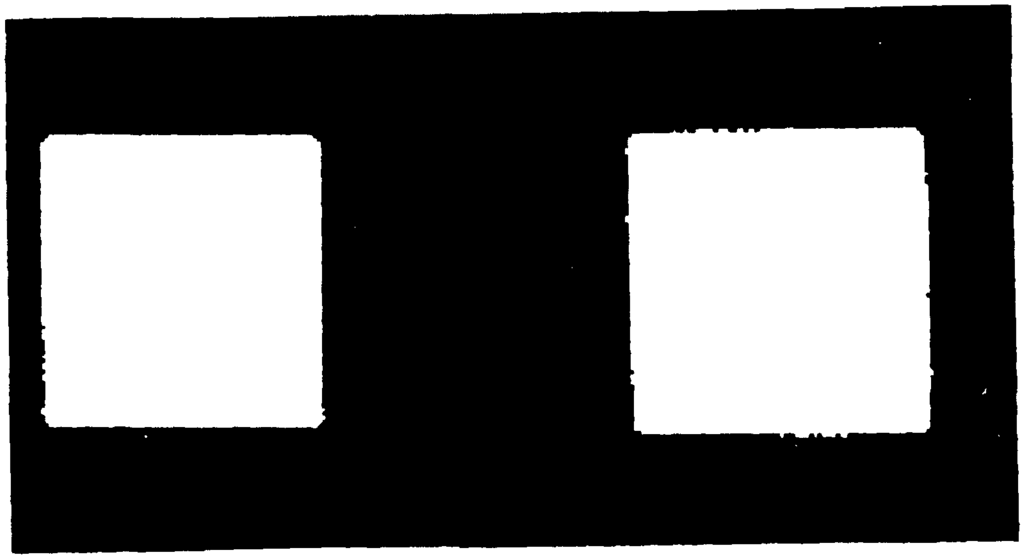

(b)

Figure 2.3 Binary image windows: (a) projected isoparametric lines, (b) camera calibration marks

domain filter, or edge detection are used to restore contrast to these parts of an image.

Each image is digitized to intensity resolution of 8 or 16 bits per pixel. The digital image is then filtered to remove noise and to enhance contrast before being thresholded into a binary image. Image noise is removed by spatial convolution with a 3 x 3 or 5 x 5 pixel operator. Contrast enhancement of the line pattern is performed by (a) frequency-domain filter, (b) local histogram modifications in small image windows, or (c) edge-detecting heuristic search [43] which is guided by the shape of the projected lines and the tendency of adjacent lines to follow similar shape. The frequency-domain filter is a band-emphasis and low-pass filter [34,60] designed to enhance the frequencies of the projected grid lines, and to remove high-frequency noise.

A program computes the coordinates of all the visible corners of camera calibration marks by taking these steps on a binary image:

(2.9)

(1) read a window of the binary image;

(2) make a fast check for a possible presence of a calibration mark in the window, if not present go to (1); (this fast check is made by parsing pixels in the $\underline{r}$ and $\underline{s}$ directions and looking for pixel strings of types 0^m1^n, 1^m0^n, or $1^m0^n1^p$;

(3) compute edge pixels of a calibration mark with an 8-point Laplacian operator;

(4) locate straight lines in edge pixels using Hough transform [23];

(5) compute parametric equations of lines fitted into these edge-lines;

(6) compute intersections of the lines; these 2D intersection points, $\overline{h}(r,s)_i$, are the corners of a camera calibration mark, go to (1).

Next, each extracted corner (r,s) is matched with its pre-stored coordinates in $O(x,y,z,w)_k$ and a five-tuplet (r,s;x,y,z) of the coordinates is formed. The camera transformation matrix $T\{O_k \rightarrow Q_i\}$ is then computed from six or more of these five-tuplets by solving the system of equations (2.5). Since the calibration stand is normally photographed by the camera only from one octant of the object coordinate space O_k, a simple relational graph [24,75] has been developed to perform the matching of the projected calibration points with their 3D coordinates. The graph contains 2D spatial relationships of the calibration marks when viewed by the camera from the given octant.

The networks of the projected parametric lines are extracted from a binary image by a program which may execute concurrently on the same image with the program extracting the camera calibration marks. This program creates a network of lines by performing these steps:

(2.10)

(1) apply a line-thinning operator to the image; this operator, similar to those given in [52,76], uses a transition table to thin iteratively lines in a binary image while preserving line connectivity, i.e., a line pixel has only 1 or 2 adjacent pixels while a node pixel has 3 or 4 adjacent pixels in an 8-pixel neighborhood;

(2) remove all pixels which do not belong to lines or nodes, i.e., pixels with 0 or more than 4 adjacent pixels (background, calibration marks, shadows, and surfaces not illuminated);

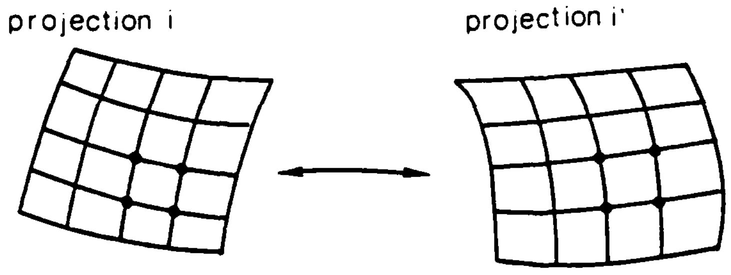
(3) convert the line pixels, still in raster format, into parametricly defined straight line segments or cubic curves; this is the scan-line-to-vector conversion where all adjacent line pixels, connecting two node pixels, are collected into clusters of $\bar{h}(r,s)_i$ points and either a 2D straight line segment or a cubic curve is fitted into them, the two end points of the line segment or the curve are also computed.

The above three steps are performed in one pass over a binary image while no more than 10 to 15 scan lines have to be accessible at the same time. After the image has been processed, node intersections are computed. For the pattern of orthogonal parametric lines used here, there are typically four line segments or curves intersecting at a node. The coordinates of a node are computed by minimizing the sum of distances to the lines or curves from the intersection point. A data structure is created for the line network where each node description contains the $\overline{h}(r,s)$ coordinates of the node, pointers to all the adjacent nodes and pointers to coefficients of curves or lines connecting adjacent nodes. Pointers of each node to its neigboring nodes are sorted in such a way that each isoparametric line forms a two-way linked list of its segments. Therefore the topology of the line network is implicitely embedded in the data structure and one may traverse the line network in any path. This capability is used during the reconstruction of the projected lines.

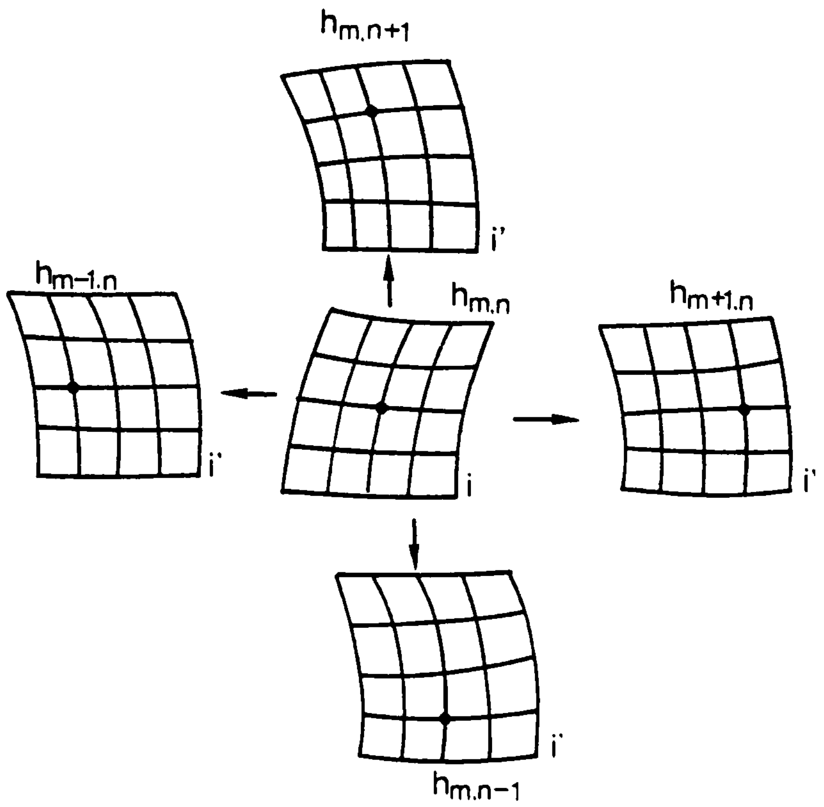
It should be emphasized that while computing the 2D projections of the surface curves, there is additional information, not needed for the reconstruction of individual 3D points, that has to be extracted from each image. When constructing cubic curves with the method of Appendix B, the slope of the projected curve at each end point must also be evaluated.

Finally, a matching algorithm which reconstructs the multiple 2D projections of the isoparametric line networks in the 3D object coordinate system is presented. This algorithm matches corresponding projections of points and curves which are then numerically reconstructed by equation (2.3) for points, and the method of Appendix B for curves. This matching is primarily based on the topology of the projected line networks. The matching proceeds in two steps: (1) an initial estimate of the match is made, and (2) the correct match is found by a gradient search.

In the first part, the algorithm locates a cycle of four adjacent nodes in one image and attempts to find projections of these nodes in the other image. The projections must also form a cycle of adjacent nodes. The projection of a node is determined when the 3D line from the center of projection to the node in the first image is projected into the second image and the node nearest to it is found. This matches node $\bar{h}_{m,n}(r,s)_i$ with node $\bar{h}_{m,n}(r,s)_{i'}$, and similarly the other three nodes [Figure 2.4(a)]. Note that the four nodes in a cycle provide the minimum surface information needed for a single patch element. This initial match succeeds if the reconstruction error, evaluated by equation (2.4), of each of the four nodes is less than a specified maximum. The acceptable maximum error depends on the resolution of the digitized images. If it fails then another cycle is located and this step is repeated; if all the cycles are exhausted without a success, then the two projections do not correspond to each other (either they contain different surfaces, or they are parametrized by differ-



(a)



(b)

Figure 2.4 Matching of projected line networks: (a) initial match, (b) final search

other (either they contain different surfaces, or they are parametrized by different parameter systems). Although we could match only a single node in this initial step, a cycle of four nodes substantially improves the match and, therefore, shortens the search in the second step.

In the second part, an evaluation function is used to search for the correct match of the two line networks. First, the current match of nodes $\bar{h}_{m,n}(r,s)_i$ and $\bar{h}_{m,n}(r,s)_{i'}$ is iteratively propagated over the line network in the directions of the parametric lines until all matchable nodes are reconstructed. Then the evaluation function which minimizes the sum of the node reconstruction errors and maximizes the number of reconstructed nodes is computed by:

$$e(m,n;m,n) = w_d \sum_{p=1}^{L} d_p + w_L/L \qquad (2.11)$$

where

d_p = reconstruction error of node p

w_d = weight of $\sum d_p$

L = number of reconstructed nodes

w_L = weight of $1/L$

e = value of evaluation function for match of node $\bar{h}_{m,n}(r,s)_i$ with node $\bar{h}_{m,n}(r,s)_{i'}$.

The first line network is then moved by one node in the four directions of the network with respect to the second network and four new matches of node $\bar{h}_{m,n}(r,s)_i$ with nodes

$s)_i$, are generated [Figure 2.4(b)]. The evaluation functions e(m,n;m+1,n), e(m,n;m-1,n), e(m,n;m,n+1), and e(m,n;m,n-1) of each match are computed, and the correct match is found when the value of the function for a match is smaller than the value at any of its neighbors, that is:

$$e(m,n;m,n) < \min\left[e(m,n;m+1,n), e(m,n;m-1,n), e(m,n;m,n+1), e(m,n;m,n-1) \right] \quad (2.12)$$

If the inequality (2.12) is not satisfied by the current match then the match with the smallest value of _e_ becomes the current match and this search is repeated until it converges on the solution and the inequality (2.12) becomes valid. The final match is then used to create a data structure which contains topological description of the reconstructed parts of the line network again defined as two-way linked lists of the orthogonal line segments and the reconstructed 3D points and curves. This data structure can be converted into a linear list of surface data needed to compute individual bicubic patches.

If the images contain two or more disconnected line networks, each network has to be processed by this algorithm separately. For more than two projections, this process is carried out for the first two projections as described and then each additional projection is matched, one at a time, with the already reconstructed network.

In many applications of surface models, specially models of closed (solid) objects, it is pertinent to distinguish between the "outside" and the "inside" sides of a surface. The "outside of a surface or object is defined as the side facing the cameras during the surface reconstruction. A surface-normal vector, pointing to the "outside" direction, is attached to each reconstructed node. The vector is, therefore, forced to be oriented so that:

$$\overline{N} \cdot \overline{H}_i > 0 \qquad (2.13)$$

where

$\overline{N}$ = surface-normal vector,

$\overline{H}_i$ = location of center of projection of image i.

The center of projection of image <u>i</u> is computed from the system of equations (2.3) using the camera transformation matrix $T\{O_k \rightarrow Q_i\}$ and two linearly independent points in the image plane. Each equation of (2.3) defines a 3D plane; their intersection is a 3D line through the center of projection. The intersection of two such lines is the location of the center of projection, denoted $\overline{H}_i$. If only two images are used to generate the 3D data, then either center of projection can be used in (2.13) to orient the normal vector since each point must have been visible in both images (otherwise it could not have been reconstructed). If more than two images are used, then each point needs a pointer to the camera transformation matrix of one of the images from which it was reconstructed. The location of the center of projection of this image is used in (2.13).

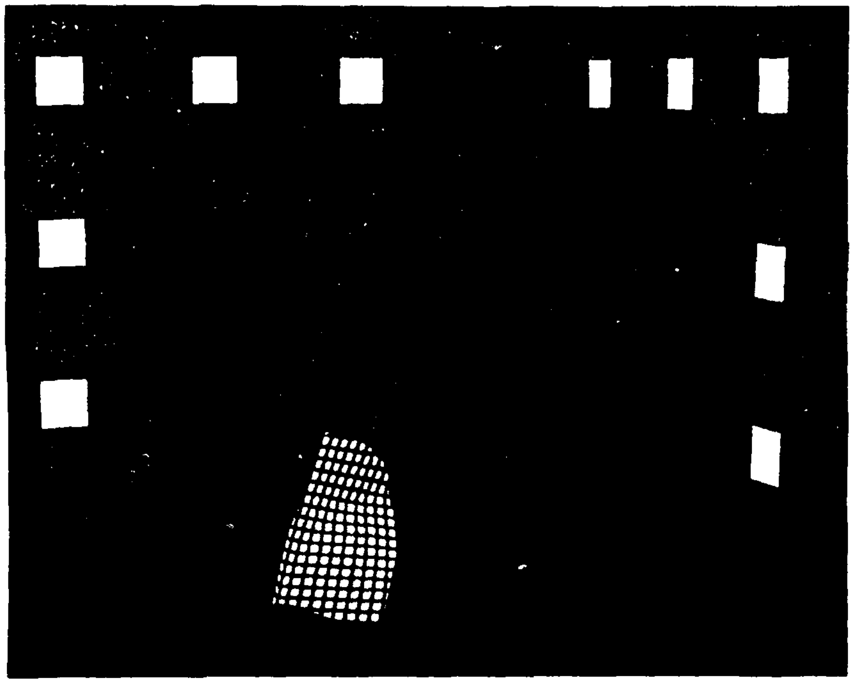
The surface information, computed at each reconstructed node, consists of the following items:

$\bar{h}(x,y,z)$	positional coordinates	(2.14)
$\bar{h}_u(x,y,z)$	surface tangent in u	
$\bar{h}_v(x,y,z)$	surface tangent in v	
$\bar{h}_{uv}(x,y,z)$	surface cross-derivative (twist)	
$\bar{N}(x,y,z)$	surface-normal vector	
pointer_1	pointer to adjacent node in +u direction	
pointer_2	pointer to adjacent node in +v direction	
pointer_3	pointer to adjacent node in -u direction	
pointer_4	pointer to adjacent node in -v direction	

The above information is used to convert the reconstructed nodes and curves into surface patches as outlined in Appendix A.

2.5 Illustrations

The surface reconstruction method is illustrated here by the following example of a multiply-curved surface. There are two images, in $Q(r,s,w)_{1,1,1}$ and $Q(r,s,w)_{2,1,1}$ image coordinate systems, of the parametrized surface as shown in Figure 2.5. The images are displayed at resolution of 1100 x 1400 binary pixels. The orthogonal parametric lines extracted from the two images are drawn, together with a


(a)

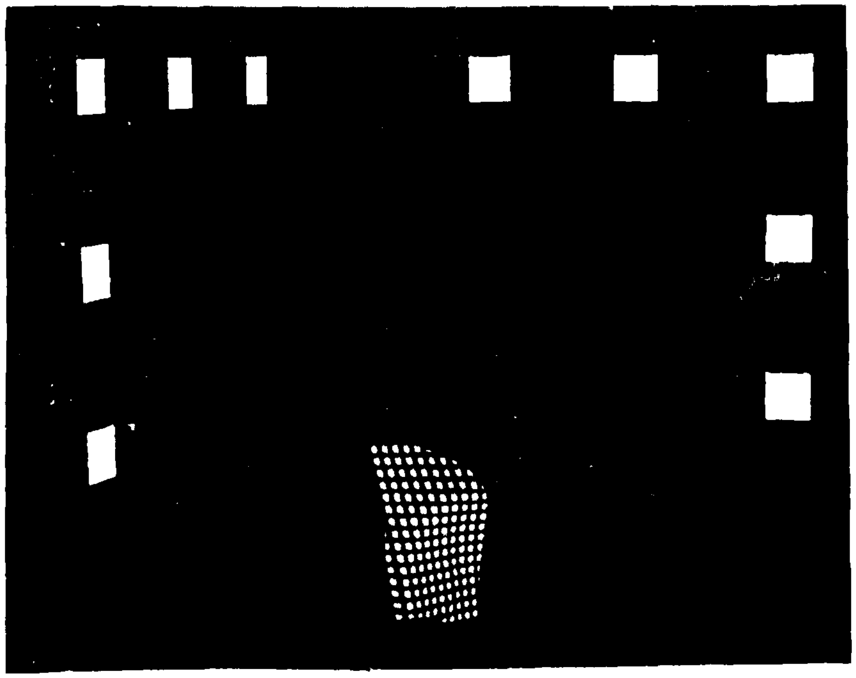
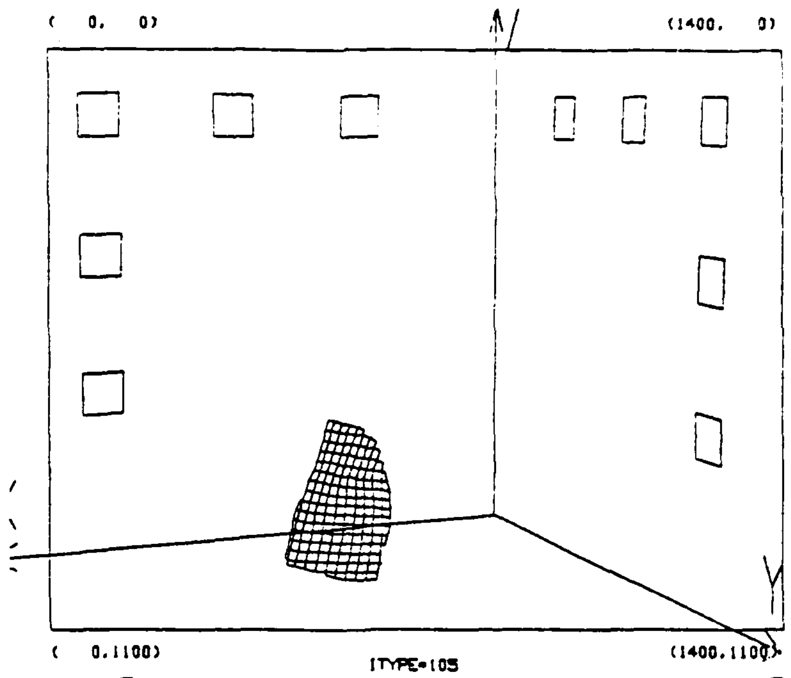

(b)

Figure 2.5 Two digitized images of a parametrized test surface: (a) $Q(r,s,w)_{1,1,1}$, (b) $Q(r,s,w)_{2,1,1}$

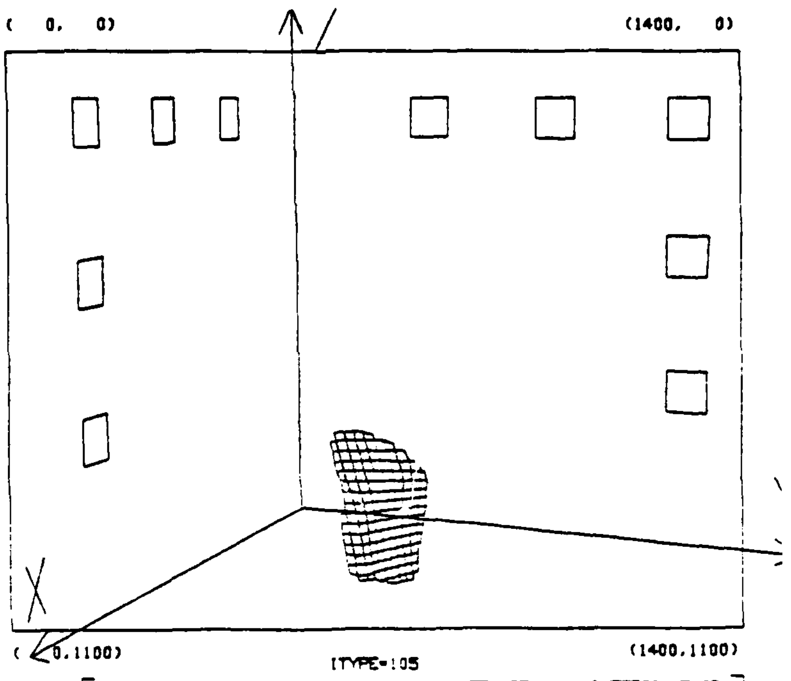
model of the camera calibration marks and the object coordinate system $O(x,y,z,w)_1$, in Figure 2.6. The image reconstruction mappings $T\{O_1 \rightarrow Q_1\}$ and $T\{O_1 \rightarrow Q_2\}$ are also printed in this figure under their corresponding images. The reconstructed surface, named 'SURFACE.3', is shown in four orthographic projections in Figure 2.7. The surface is represented by a network of 3D polygons. There are four different projections of the surface shown in Figure 2.8. Here, the surface is represented by a sheet of composite 3D bicubic patches with C^1 continuity. Shaded synthetic images of the sheet of patches are shown in Figure 2.9.

2.6 Summary

A technique for reconstruction of 3D surface information from multiple images was presented in this chapter. It is based on correlation of projected patterns of light on the surfaces being measured. Triangulation is used to compute 3D surface data from 2D data obtained in two or more images. Image transformations are computed from a number of calibration marks whose locations in the 3D space are known, and whose locations in an image are measured. Arbitrary camera positions and orientations are therefore permitted, however, limitations of stereo correlation still persist: reconstruction error increases with narrow separation angle, reconstruction data decrease with wide separation angle.



$$T = \begin{bmatrix} -0.2381874E+01 & 0.4964848E+00 & -0.3946236E-01 & 0.8473995E+03 \\ -0.4389674E-01 & -0.6297228E-01 & -0.2407152E+01 & 0.8829127E+03 \\ -0.2638782E-03 & -0.6275908E-03 & -0.7184327E-04 & 0.1000000E+01 \end{bmatrix} \quad \text{(a)}$$



$$T = \begin{bmatrix} -0.1328022E+01 & 0.1996885E+01 & -0.7155143E-01 & 0.5556011E+03 \\ -0.5607650E-01 & -0.2392324E-01 & -0.2375292E+01 & 0.8701680E+03 \\ -0.6574546E-03 & -0.2646891E-03 & -0.7571734E-04 & 0.1000000E+01 \end{bmatrix} \quad \text{(b)}$$

Figure 2.6 Information obtained from images in Figure 2.5: (a) $Q(r,s,w)_{1,1,1}$, (b) $Q(r,s,w)_{2,1,1}$

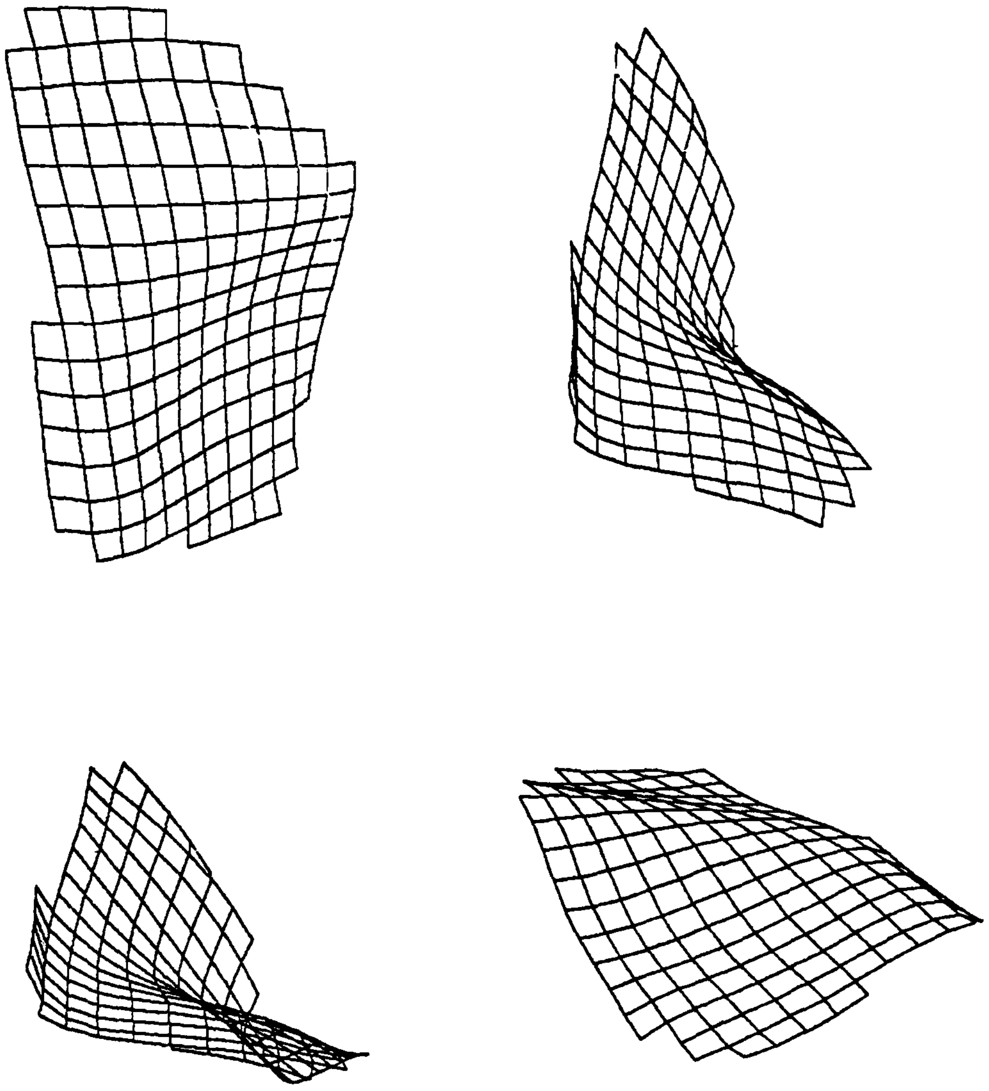


Figure 2.7   Four views of the reconstructed surface described by a 3D network of isoparametric lines

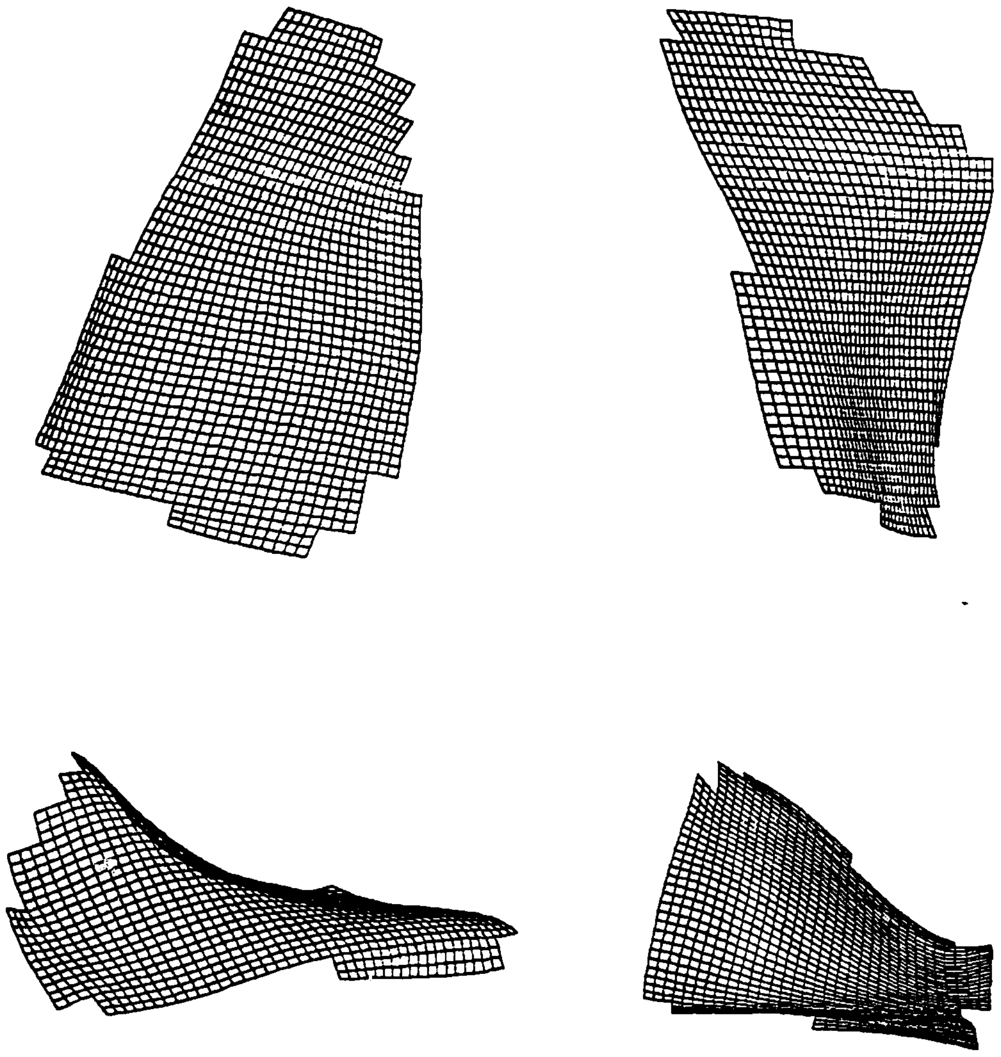


Figure 2.8 Four views of the reconstructed surface described by a sheet of bicubic patches

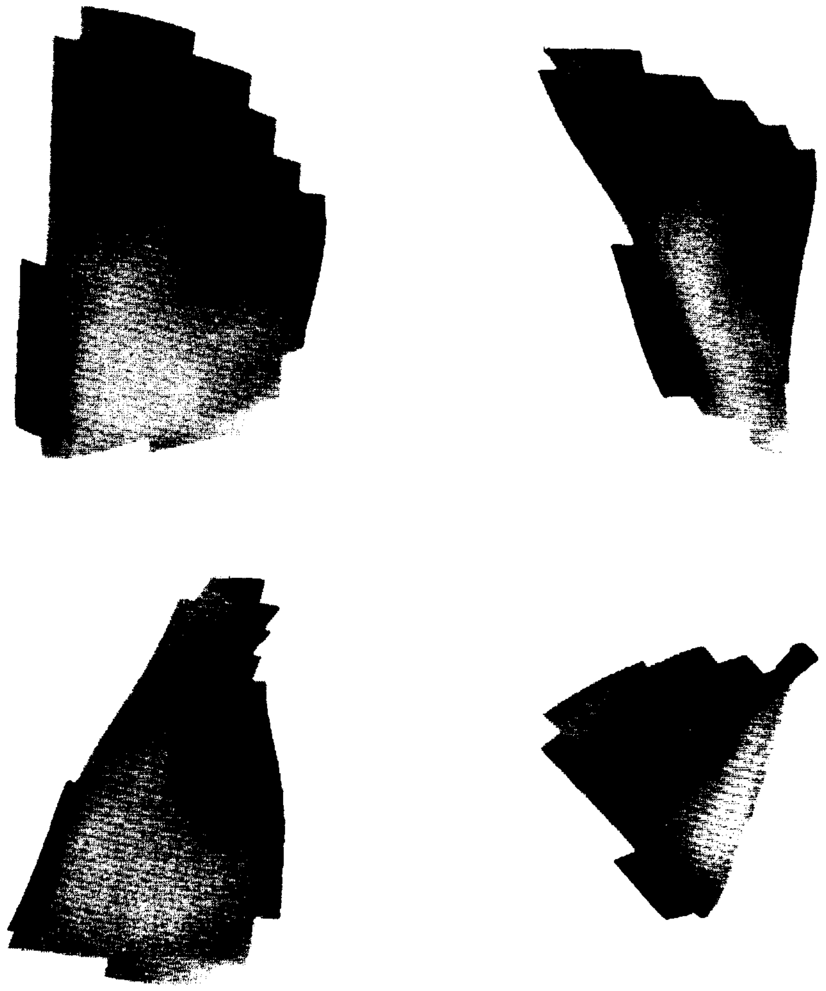


Figure 2.9 Four synthetic images of the reconstructed surface described by a sheet of bicubic patches

This method allows additional images to be used to improve accuracy of the measured surfaces and to increase the measured surface area. However, all surface measurements in this work were obtained from pairs of images sequentially taken by the same camera.

CHAPTER 3

HIERARCHICAL SURFACE REPRESENTATIONS

Computer surface modeling refers to the ability to analyze a three-dimensional surface and represent it by a composite description based on many independent features such as size, shape, structure, texture, and light reflective and transmissive characteristics. The creation of such a computer model is necessary for any computer vision or scene analysis system. This chapter describes a surface and volume representation system which was developed to support the modeling process of this work.

The representation of 3D surfaces in a data base is hierarchical; that is, it consists of a control structure which in the most general case is a directed, acyclic graph. The nodes of the graph contain descriptions of the surface shapes and their relations, and the arcs represent 3D rigid transformations in the object coordinate system. There are three types of nodes in such a graph: (1) surface elements which contain the actual surface geometry, (2) bounding volumes which partitions the 3D object space to facilitate efficient searching and sorting, and (3) surface relationships which contain logical connections among surface and volume sections. Although all three types of nodes are applicable to geometrical modeling found in CAD/CAM and computer graphics, the last type is also useful for object matching and recognition in computer vision applications. The system described here also permits volume (or solid) modeling as well as surface modeling. Solid primitives are designed from surface elements which enclose 3D volumes. Complicated solid objects are constructed by combining the solid primitives with set operators. This allows the system to be also used in _constructive solid geometry_ (CSG) applications [4,62].

The processing algorithms which operate on this surface representation are based on the method of ray-tracing. This method requires the ability to traverse the data structure and compute intersections between a 3D line (a ray) and 3D surface elements. It is an effective, versatile, but rather brute-force method which has been used for the generation of shaded images [72,64,63] and line drawings [63], computations of mass property [62,63], and conversions of representation [62,63]. In the next chapter we develop algorithms for (a) matching of 3D surface descriptions which use ray-tracing to evaluate a measure of shape similarity between two surfaces, and (b) merging of overlapping surface descriptions which use ray-tracing to compute parameter transformation, parameter projection, and conversion of representation.

In addition to using the surface information provided by the photogrammetric method described in the previous chapter, surface data obtained from other sources or synthetically designed may also be entered into this surface data base. The section of Chapter 3 which most directly applies to the modeling process is the conversion of the surface information generated by the reconstruction method into surface quadtrees which can then be passed to the various processing algorithms. Chapter 3 also reviews other forms of surface elements, in addition to parametric patches, that are used in the modeling system described here, and basic geometric operations on these elements required by the processing algorithms.

3.1 Graph and Tree Representations

The hierarchical surface representations are stored in directed, acyclic graphs. A directed graph is generally defined to be a finite, non-empty set of nodes together with a set of directed arcs joining pairs of distinct initial and final nodes. An acyclic graph does not contain any cycles of arcs. It has one node of in-degree 0 which is referred to as the root node. The final nodes of arcs leaving a node are called its successors; similarly, the initial nodes of arcs coming to a node are called its predecessors. A tree is an acyclic, directed graph in which all nodes except the root node have in-degree of exactly 1. In a hierarchical graph or tree, a node contains more detailed information than any of its predecessors. In an ordered graph or tree, the successors of a node are listed in a determined order.

The data structure of the surface representation, as shown in Figures 3.1 and 3.2, consists of the graph and a number of tables which provide additional information about each node or arc. The graph contains three types of nodes, called the R-nodes, B-nodes, and E-nodes, which describe surface relations, bounding volumes, and surface elements, respectively. Although the nodes in the graph are independent of the surface representation, there is a set of separate tables for each type of node. The actual information about surface relations, bounding volumes, and surface elements is stored in sets of R-tables, B-tables, and E-tables, respectively. Detailed descriptions of the contents of these tables are given in the next three sections of this chapter. The 3D transformations represented by the arcs of the graph are stored in the T-table. A null pointer from an arc implies identity transformation. A node entry in the graph itself contains the information shown in Figure 3.2, and is stored in a variable-length block in the G-table. This information consists of (1) the type of the node, (2) the number of its table and a pointer to an entry in it, (3) three flags which determine whether the successors of the node represent a closed volume or an open surface, whether they are to be treated as actual surfaces and volumes or as invisible, auxiliary surfaces and volumes, and whether they contain a complete object or a collection of surfaces and

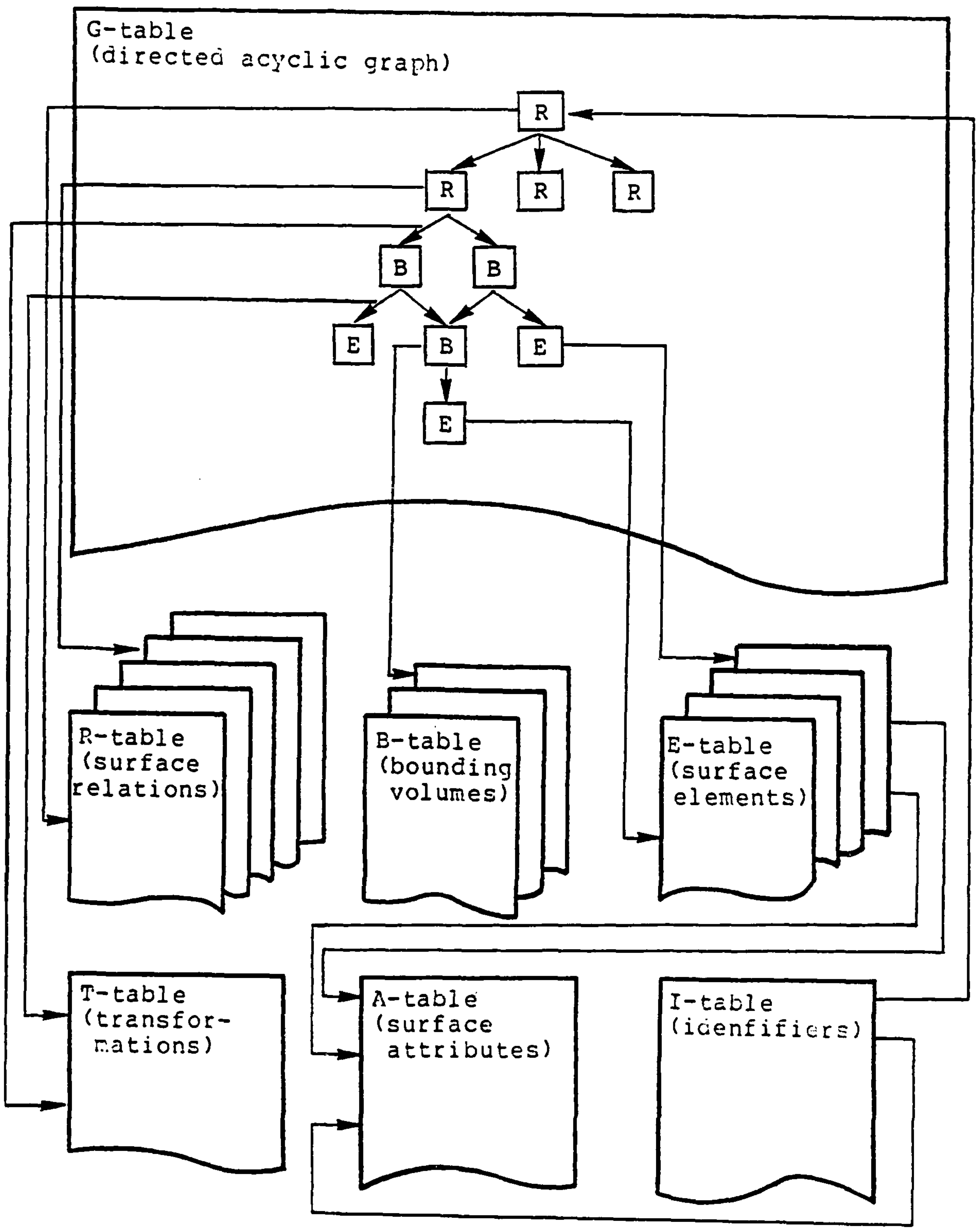
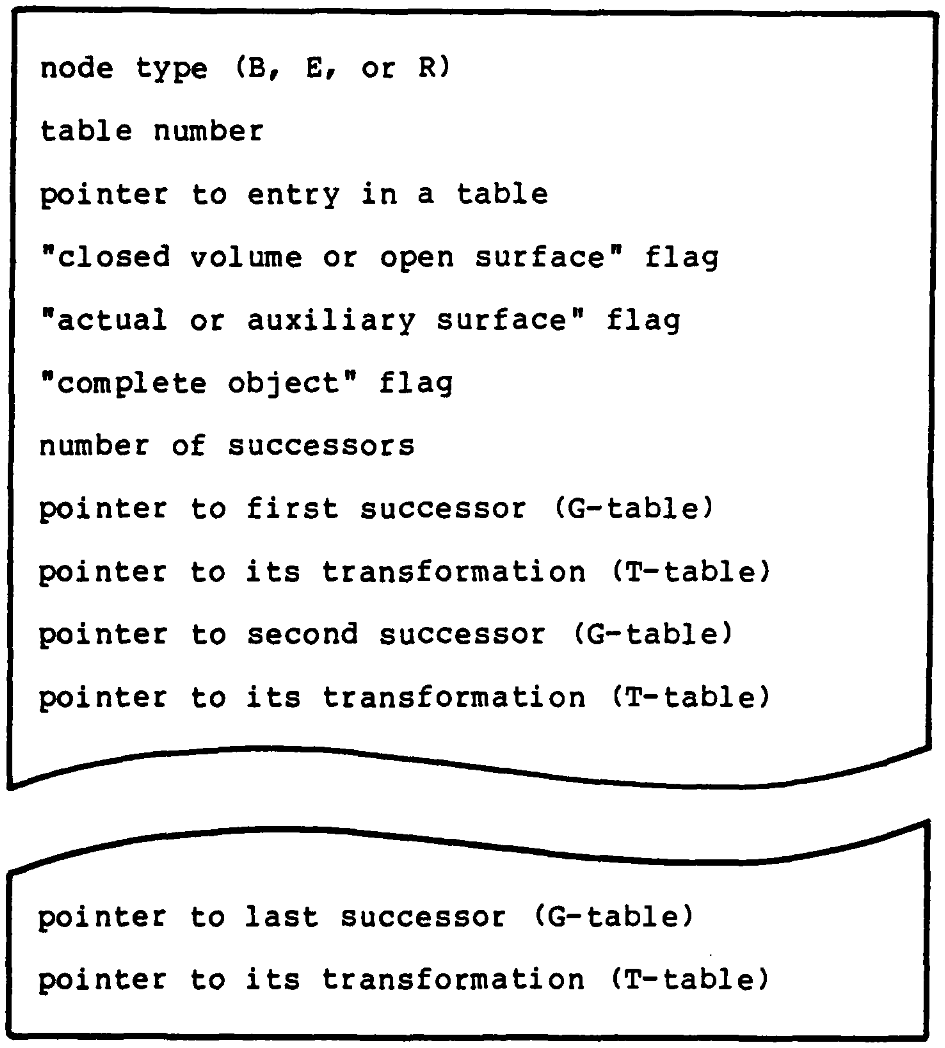


Figure 3.1 Diagram of hierarchical surface representation



node type (B, E, or R)

table number

pointer to entry in a table

"closed volume or open surface" flag

"actual or auxiliary surface" flag

"complete object" flag

number of successors

pointer to first successor (G-table)

pointer to its transformation (T-table)

pointer to second successor (G-table)

pointer to its transformation (T-table)

pointer to last successor (G-table)

pointer to its transformation (T-table)

Figure 3.2 Contents of a graph node

volumes, (4) the number of successors, and (5) an ordered list of pointers to the successors, each of which has an associated pointer to a 3D transformation. Note that each bounding volume or surface element may potentially be defined in its own local 3D coordinate system while the entire graph is defined in a single global object coordinate system O(x,y,z,w). Additionally, the data structure consists of the A-table which contains surface attributes of individual surface elements, and the I-table which contains identifiers to allow symbolic references to any entry in any table. A summary of the graph representation is given in Figure 3.3.

Two special cases of tree representation need to be mentioned now: quadtree and octree. A quadtree is a hierarchical, ordered tree in which each non-terminal node has an out-degree of 4. It has been used to encode efficiently information formatted on a 2D grid, such as digital images [34,62], and, therefore, it can also store bivariate surface representation as will be shown in Section 3.6. Similarly, an octree is a hierarchical, ordered tree in which each non-terminal node has an out-degree of 8. It is being used to encode and process efficiently spatial information formatted in a non-homogeneous 3D lattice [41].

The graph representation employed here has the following main advantage over a tree representation: it allows a part of the surface representation which is duplicated in the object space to be stored only once, and

Node	Contents	Type
B	bounding volume	
		sphere
		ellipsoid
		parallelepiped
E	surface element	
		plane
		sphere
		quadric surface
		bicubic patch
R	surface relationship	
		shape classification
		alternative representation
		logical operation
		object decomposition
		object partition

Arc	Contents
T	transformation

Table	Contents
A	surface attributes
B	bounding volumes
E	surface elements
G	surface graph
I	symbolic identifiers
R	surface relationships
T	transformations

Figure 3.3 Summary of hierarchical surface representation

each of its instances to be generated with the appropriate 3D transformation during a traversal of the graph. Similarly, each volume primitive needs to be defined only once in a standard orientation and then transformed into its actual locations in a solid object.

Either depth-first or breadth-first traversal of such a graph is possible. Depth-first traversal is used in the processing algorithms here. It requires a stack in which the pointers to all successors of the current node, which are to be visited, are placed together with their 3D transformations. These transformations are computed by concatenating the 3D transformation of the current node with the transformation in the arc pointing to the successor. The next node on the top of the stack is visited next until the stack is emptied. However, if a program frequently traverses a graph, these redundant computations of 3D transformations can be eliminated by taking a pre-processing step which (1) transforms all the coordinate information in the graph to the absolute object coordinate system, and (2) converts the graph structure to a tree structure, i.e., it generates all instances of nodes with in-degree greater than 1.

3.2 Relational Connections

The relational connectivity of a surface model gives the logical and geometrical relations among the parts of a surface, among the parts of an object, or among a group of objects. These relations are mainly intended for any scene analysis and object recognition work that may be performed with this representation. The following five relational categories are currently defined:

(1) shape classification,

(2) alternative representations,

(3) logical operations,

(4) object decomposition, and

(5) object partitions.

The relational connectivity of a surface is stored in the R-nodes of the surface graph and in the associated R-tables.

The shape-classification table contains general information on the 3D shape of the first successor of the current R-node. It is classified into one of these standard shapes: stick, blob, or box. The second successor contains a polyhedral convex-hull representation.

The alternative-representation table provides the ability to switch from one representation to another; therefore, the same surface may be described by more than one representation. The first successor of the current R-node contains the original representation; the subsequent successors contain the alternative representations which usually are polyhedral or quadric surface approximations.

The logical-operation table defines logical relations of the successors of the current R-node. At the present time, there are defined three set operators: <u>union</u>, <u>intersection</u>, and <u>difference</u>, which operate only on <u>closed</u> volumes. The union operator produces the union of all the successors. The intersection operator produces the intersection of the first successor with each subsequent successor. The difference operator produces the difference between the first successor and each subsequent successor. Note also that there is an implied union of all successors in each B-node. During the ray-tracing traversal of a graph, all the intersections of the ray and the volumes contained in the successor nodes are combined according to the specified set operator into line segments of the ray inside the valid volumes [63]. These set operators are used in a number of solid modeling systems [4,62,63] to design complicated objects with CSG trees of volume primitives. They are specially useful for interference checking and sectioning. In the next chapter, we will propose a segmentation algorithm that will convert an arbitrary 3D solid object into a CSG tree made of these three operators and a set of solid primitives by employing the surface matching algorithm.

The <u>object-decomposition</u> table allows logically and physically separable parts of an object to be segmented, and their spatial relationships, such as "on top of," "next to," "supports," "attaches," "connects," "screws in," etc., to be identified. A list of relationships between all pairs of successors is provided. Such segmentation is essential for relational matching of 3D objects [46,69].

The object-partitions table provides the means to group a set of objects stored within the same graph into partitionss of similar objects. A successor of the current R-node contains either an object or a group of objects with similar shape. The last successor of this node then contains a characteristic description of the objects in the preceding successors, which typically is a coarse polygonal approximation. When a graph containing a library of objects is used to identify an unknown object, the unknown object is matched against the objects in the node's successors only if the unknown object is close to the object description in the current node [69].

3.3 Bounding Volumes

A bounding volume is an "invisible" surface element or a set of surface elements completely enclosing a volume which in turn may contain other bounding volumes or surface elements. The purpose of bounding volumes is to partitions the 3D object space to minimize the searching and sorting of surface elements as is required in a number of algorithms for processing 3D surfaces. During the ray-tracing traversal of a graph, whenever a ray misses a bounding volume, it also misses the bounding volumes and surface elements within it and, therefore, the successors of the bounding volume are not visited. An intersection between a ray and a bounding volume is usually much faster to compute than an intersection between a ray and a surface element. A bounding volume may optionally contain its own, appropriately coarse, description of the surface elements within it, usually computed by merging these surface elements. Should a processing algorithm determine, from, let us say, the size of the bounding volume that a more precise description of the surface is not required, it uses the surface description in the bounding volume and omits to visit the bounding volume's successors. Possible bounding volumes are:

(1) sphere,

(2) ellipsoid, and

(3) parallelepiped.

Ellipsoids and parallelpipeds should be oriented to minimize their volume. Only spherical bounding volumes are used in the examples throughout this work.

A bounding volume is stored in a B-node of the graph. It contains the type of the volume and a pointer to an entry in a B-table. A B-table contains the coefficients of the volume and an optional approximation of the surface elements within the volume. There is also a normal vector which is the average of the normal vector field to the surface elements in the terminal nodes within the volume. The magnitude of each normal vector is proportional to the surface area it represents. This, in effect, gives a hierarchical representation of the surface's Gaussian Map [22].

A bounding sphere is computed by an algorithm for the problem: "given a set of N points in 3D space, find the smallest sphere enclosing them." The smallest enclosing sphere is defined either by two points which define its diameter or by three points of the set. An algorithm which tests all the spheres defined by two or three points of the set and selects the smallest sphere which encloses all the points runs in $O(N^2)$ time. An improved algorithm by Shamos [64] locates the defining points of the smallest sphere from the extreme points of the set by constructing a Voronoi diagram and runs in $O(N \log N)$ time.

A bounding volume entry contains the following information stored in a B-table:

(1) a description of the bounding volume geometry;

(2) an optional approximation to the surface contained within the volume;

(3) error of this approximation, computed as the average distance between the actual surface contained in the the terminal nodes and the current approximation; and

(4) average surface-normal vector integrated over the actual surface within the bounding volume.

The error between an actual surface and its approximation in a bounding volume is also computed by the ray-tracing method. Rays from the actual surface representation in the surface-normal direction are intersected with the approximated representation. The error is computed as the average of the intersected distances, each weighted by the surface area which the ray represents.

3.4 Surface Elements

A surface element is a geometric representation of the surface shape. Surface elements are contained in the terminal E-nodes of the graph representation. The surface elements used here are:

(1) planar face,

(2) sphere,

(3) quadric surface, and

(4) bicubic patch.

The surface elements usually have two different representations - algebraic and parametric. The algebraic representation is an equation in the form $f(x,y,z,w) = 0$. This representation, which is used in algebraic geometry, is useful in tasks such as "determine whether a given point $\overline{h}(x,y,z,w)$ is on the surface, inside, or outside." The bounds of the actual surface given in this representation are other algebraic surfaces in structured logical trees. For a given point to be a part of the actual surface, it must satisfy the logical relations (inside, outside) given by the bounds. The parametric representation is a vector $\overline{g}(u,v) = [\ x(u,v),$

y(u,v), z(u,v), w(u,v)] where each component is an equation of the two parameters _u_ and _v_. This representation, which is used in differential geometry [22], is useful in tasks such as "generate all points on the given surface." The bounds of a surface element in this representation are limits on the parameters _u_ and _v_.

An E-node contains the type of the surface element and a pointer to an entry in an _E-table_ which contains the coefficients of the surface representation. The two representations are given in detail for each type of surface element in Appendix C. There is a separate E-table for each type of surface element. The surface bounds are defined in additional auxiliary tables. The planar elements have a table of edges and a table of vertices; the quadric elements have a table of logical trees of other quadric surfaces. Sphere representations do not have bounds, and bicubic patches are defined over a unit square of the parameter space. Each surface element also has a set of surface attributes containing various properties of the surface other than its shape. These are stored in the _A-table_. Currently, only reflective, transmissive, and texture properties are meaningful for they can be displayed in shaded synthetic images.

A surface element in the algebraic representation defines two half spaces - inside and outside. The implied union operator present in each bounding-volume node can be used to combine the inside half spaces into a closed volume which then can serve as a volume primitive. A closed volume can be defined with the parametric representation by endless variations on the isoparametric brick theme.

3.5 Geometric Operations

There are five elementary geometric operations that need to be performed on the surface elements and the bounding volumes:

(1) 3D rigid transformations,

(2) evaluation of surface-normal vectors,

(3) evaluation of surface curvatures,

(4) intersection with 3D lines, and

(5) test for surface existence.

A 3D rigid transformation $T\{O_k \rightarrow O_{k'}\}$ of a surface element or a bounding volume from object space O_k to object space $O_{k'}$ is specified by a transformation matrix (C.3) or by a list of x-y-z displacement, rotation, and scale parameters (C.4) from which this matrix is computed (C.5-8). An entry in the T-table contains an optional list of the parameters and the transformation matrix $T\{O_k \rightarrow O_{k'}\}$ as well as its inverse $T\{O_{k'} \rightarrow O_k\}$. The 3D transformation is computed during traversal of a surface graph by concatenating transformations stored in the graph's arcs until a bounding volume or a surface element is reached.

The normal vector to a surface element or a bounding volume is computed from the algebraic representation by equation (C.10), or from the parametric representation by equation (C.11). The normal-surface curvature of the parametric representation is computed by (C.12).

The intersection between a surface element or a bounding volume and a 3D line, specified in parametric form by (C.13), is computed by substituting the parametric line representation into the algebraic representation of the surface element and solving the general equation $f(x(t), y(t), z(t), w(t)) = 0$ for $\underline{t}$. Note that the intersection of a line and a bounding volume usually does not have to be solved exactly, it is sufficient to determine whether the line pierces or misses the bounding volume. The intersections of a parametic patch, which does not have an algebraic representation, and a 3D line is computed by solving a system of three simultaneous cubic equations (C.34) with the parameters of the line ($\underline{t}$), and the patch ($\underline{u}$,$\underline{v}$) as the unknowns. The algorithm which solves (C.34) is given in (C.41). Alternatively, this problem can be presented as that of computing the intersection of a line represented by the intersection of two 3D planes with the patch which is a similarly complex problem of solving two simultaneous cubic equations with patch parameters ($\underline{u}$,$\underline{v}$) as the unknowns. All the results which involve the computation of an intersection between a parametric patch and a line in this report, either in surface matching and merging or in image generation, have been computed with algorithm (C.41).

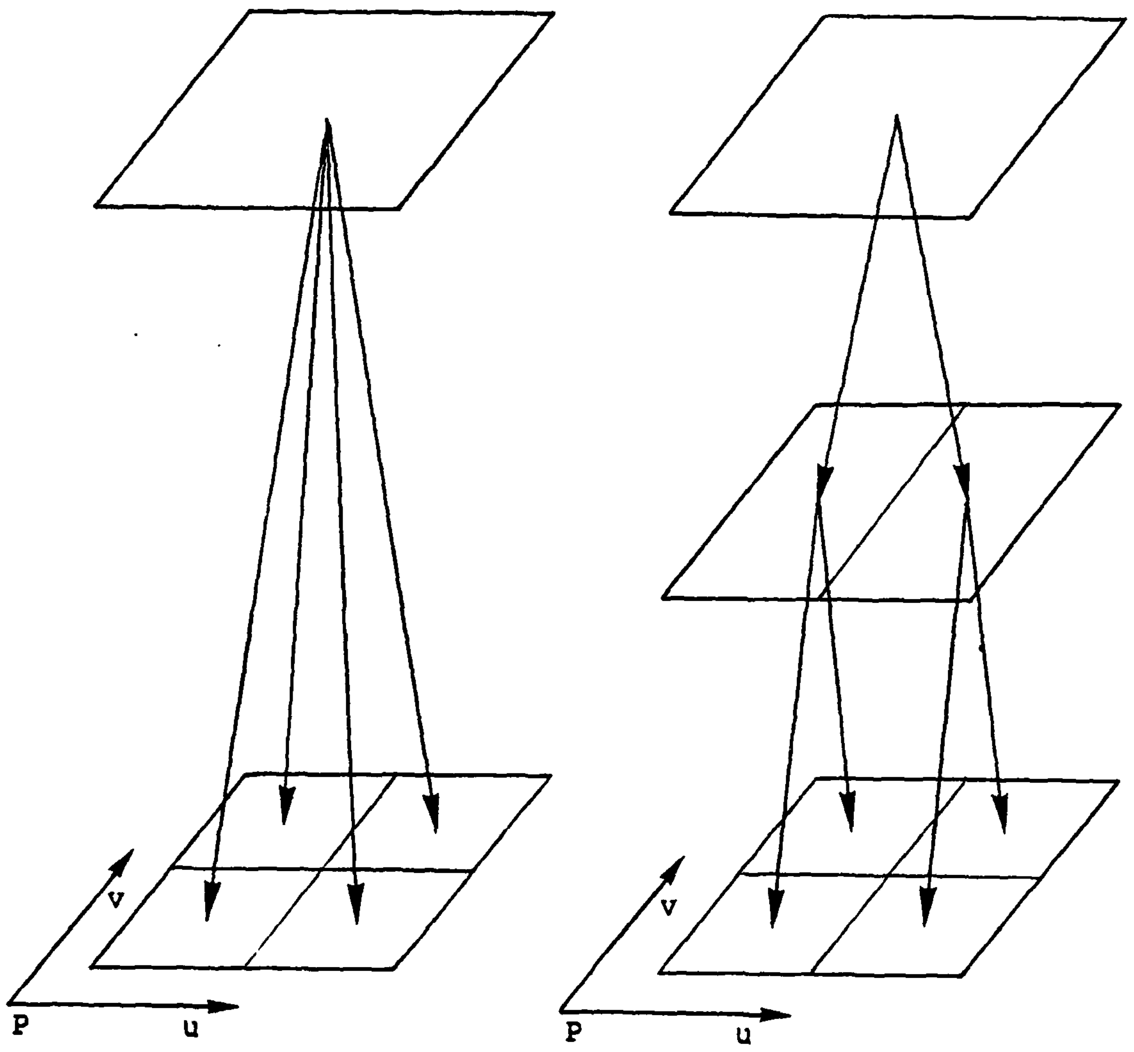
The intersection evaluation is always followed by a check to determine whether the intersection point is a valid surface point, i.e., whether it is within the specified surface bounds. For algebraic representations this involves searching logical structures of bounding surfaces also given in the algebraic form. For parametric representations the $\underline{u}$, $\underline{v}$ parameters of the intersection point must be within the designated limits of the parameters. Details describing all these geometric operations for each type of surface element are given in Appendix C.

3.6 Generation of Surface Quadtrees

This section describes a recursive algorithm which computes a quadtree structure of 3D bivariate surface data formatted in a 2D array. The array is defined by a 2D parametric coordinate system P(u,v,w). We assume that we are given an ($\underline{M}$+1) x ($\underline{N}$+1) array of 3D control points and, optionally, also 3D curves connecting the adjacent points. Some of the points and curves in the array may be missing. A surface patch can be computed from a cycle of four adjacent points $\bar{h}_{m,n}(x,y,z,w)$, $\bar{h}_{m+1,n}(x,y,z,w)$, $\bar{h}_{m+1,n+1}(x,y,z,w)$, and $\bar{h}_{m,n+1}(x,y,z,w)$; similarly, it can be computed from a cycle of four adjacent boundary curves. The details of these procedures are given in Appendix A. The quadtree algorithm uses the full resolution of the array to generate surface elements (patc'es) and bounding volumes (spheres) at the bottom level of the quadtree. It then passes the data points at the intersections of every other row and column of the array to the next level of the quadtree; that is, it reduces the resolution of the array by a factor of two in each direction. The next level is computed from this reduced resolution, linked to the level below it, and the process is recursively repeated until the array is reduced to 2 x 2 data points and the root of the tree is reached.

Once the coefficients of a patch are computed, a grid of 3D points within the patch is generated. The points are used to compute (a) the bounding volume (a sphere), and (b) a discrete field of normal vectors [22], which is averaged into a single normal vector representing the orientation of the patch. Each normal vector is weighted by the surface area it represents.

Four patches which share a common control point are merged into a single patch at the next higher level of the quadtree, as shown in Figure 3.4. The successors of a non-terminal node in a quadtree are always ordered, relative to the parametric coordinate system P(u,v,w), as is also shown in this figure. Because we need not only the 3D positional data to compute surface patches, but also the 3D slope data [Appendix A], we actually use the points in a reduced array to define positions, and the deleted points to



Quadtree

Binary tree

Figure 3.4 Quadtree and binary tree structures of bivariate surface representation in a parametric coordinate system P(u,v,w)

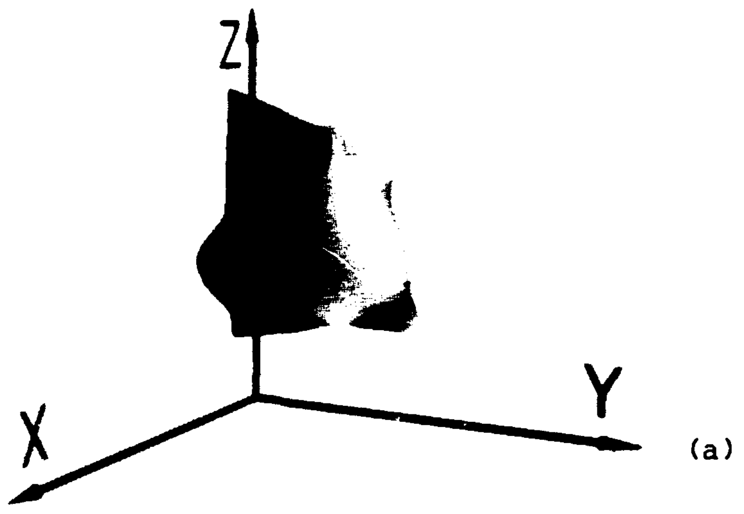
help in estimating slopes. If not all four patches are defined, then only the bounding volume is computed at the next level. When computing bounding volumes at levels other than the bottom level, care must be taken that each volume encloses not only the actual surface elements but also the surface approximations in bounding volumes within the current one.

An example of the quadtree construction procedure given in Figure 3.5(a) shows a sheet of 4 x 4 patches computed from an array of 5 x 5 control points. Figure 3.5(b) shows several bounding volumes in the quadtree as translucent spheres. The four smallest spheres - with green tint - are at the bottom level of the quadtree, each cast around a patch. There are 12 more such spheres, but which are not shown in this image. The medium size spheres - with blue tint - are in the middle level of the quadtree, each cast around 2 x 2 patches. Finally, the largest sphere - with red tint - is at the top level of the quadtree, cast around all 4 x 4 patches.

An outline of the algorithm which recursively builds the quadtree structure from a sheet of surface data is as follows:

(3.1)

```
procedure quadtree(level,M,N);
   begin
   if level = 0 then
      begin
```



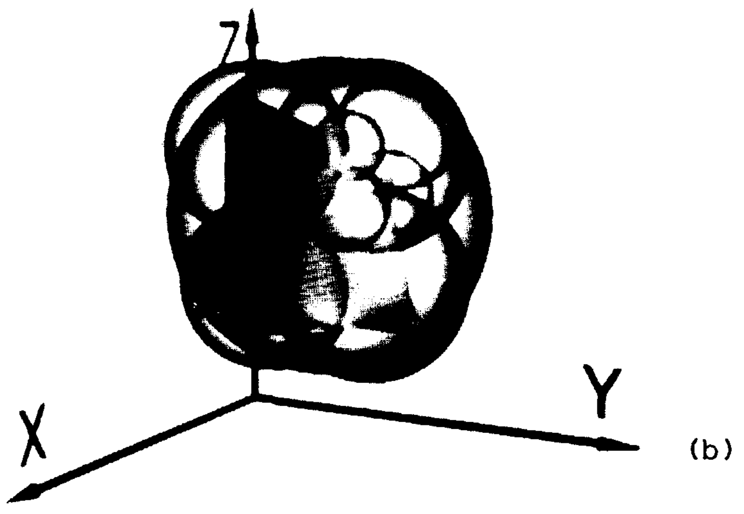


Figure 3.5 Image of a 4 x 4 sheet of bicubic patches in an object coordinate system O(x,y,z,w): (a) only surface elements shown, (b) surface elements and bounding volumes of a quadtree shown

```
          for m := 0 step 1 until M do
             begin
             for n := 0 step 1 until N do
                begin
                if surface data at (m,n) exists then
                   begin
                   compute surface element (m,n);
                   create E-node at (level,m,n);
                   compute bounding volume (m,n);
                   create B-node at (level,m,n);
                   end;
                end;
             end;
          end;
       else
          begin
          for m := 0 step 2 until M do
             begin
             for n := 0 step 2 until N do
                begin
                if B-node defined at
                  (level-1,m,n) | (level-1,m+1,n) |
                  (level-1,m,n+1) | (level-1,m+1,n+1)
                then
                   begin
                   merge bounding volumes of defined B-nodes
                     at (level-1,m,n), (level-1,m+1,n),
                     (level-1,m,n+1), and (level-1,m+1,n+1);
                   if all 4 B-nodes exist then
                      begin
                      compute surface approximation;
                      compute error of approximation;
                      end;
                   create B-node at (level,m/2,n/2);
                   end;
                end;
             end;
             M := M/2;
             N := N/2;
          end;
       level := level + 1;
       if M>2 | N>2 then quadtree(level,M,N);
       return;
       end;
```

An alternative arrangement to the surface quadtree representation is the binary tree representation. Such a tree is created by merging two adjacent surface elements which share a common boundary at any level of the tree [Figure 3.4]. The direction of the boundary across which the elements are merged usually alternates from level to level of the tree. The main goal, however, when selecting the direction of a merger is to minimize overlap of the bounding volumes.

3.7 Illustrations

The example of the reconstructed test surface 'SURFACE.3' from Chapter 1 is continued here. Figure 3.6 shows the quadtree structure of the surface; the branches of the tree are solid and the boundary curves of the patches are dotted. The nodes of the quadtree are the locations of the centers of the bounding spheres. Figure 3.7 shows the surface-normal vector at each node of the quadtree; the branches of the tree and the patch boundary curves are dotted and the normal vectors are solid. This model, 'SURFACE.3', consists of the following information:

1	sheet
207	bounding volumes
176	control points
146	patches at level 0
5	quadtree levels

The surface shown in Figures 3.6 and 3.7 is contained in the bottom level of the quadtree. Also, the four images in Figure 2.11 were generated from the bottom level of the quadtree.

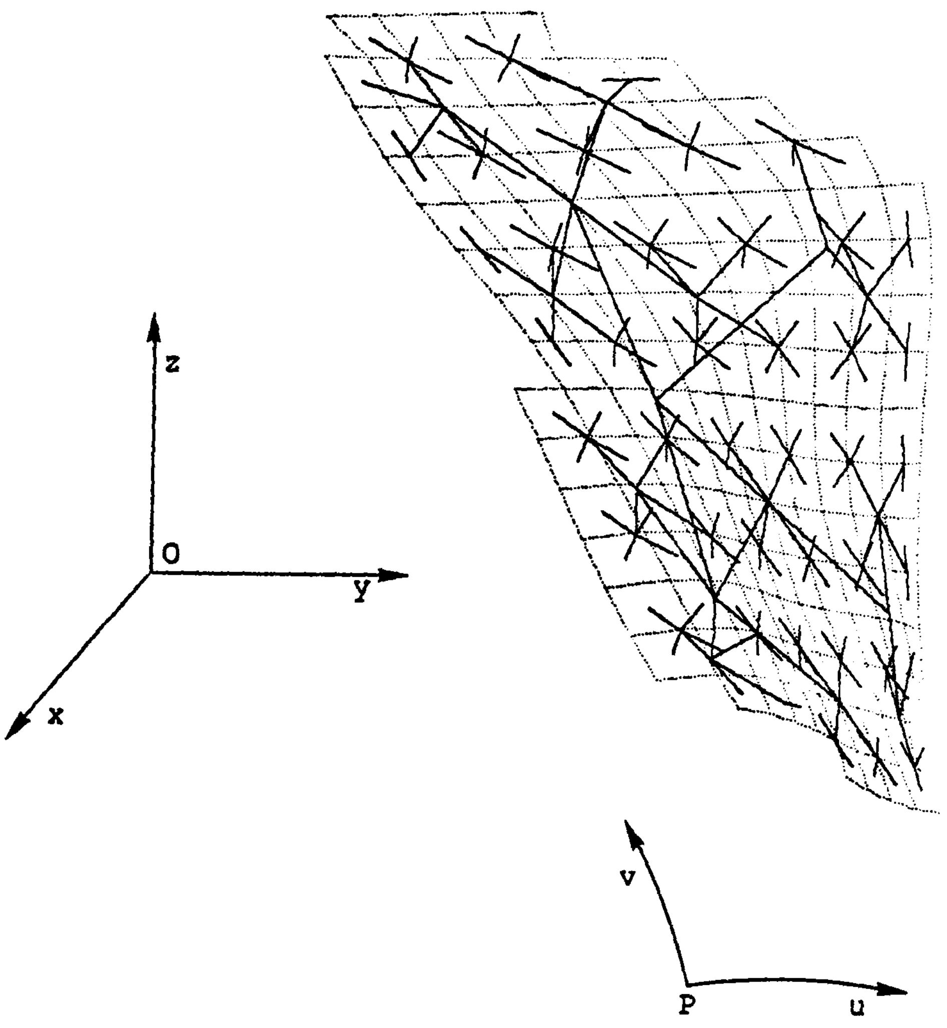


Figure 3.6 Quadtree of surface patches (tree branches solid, patches dotted)

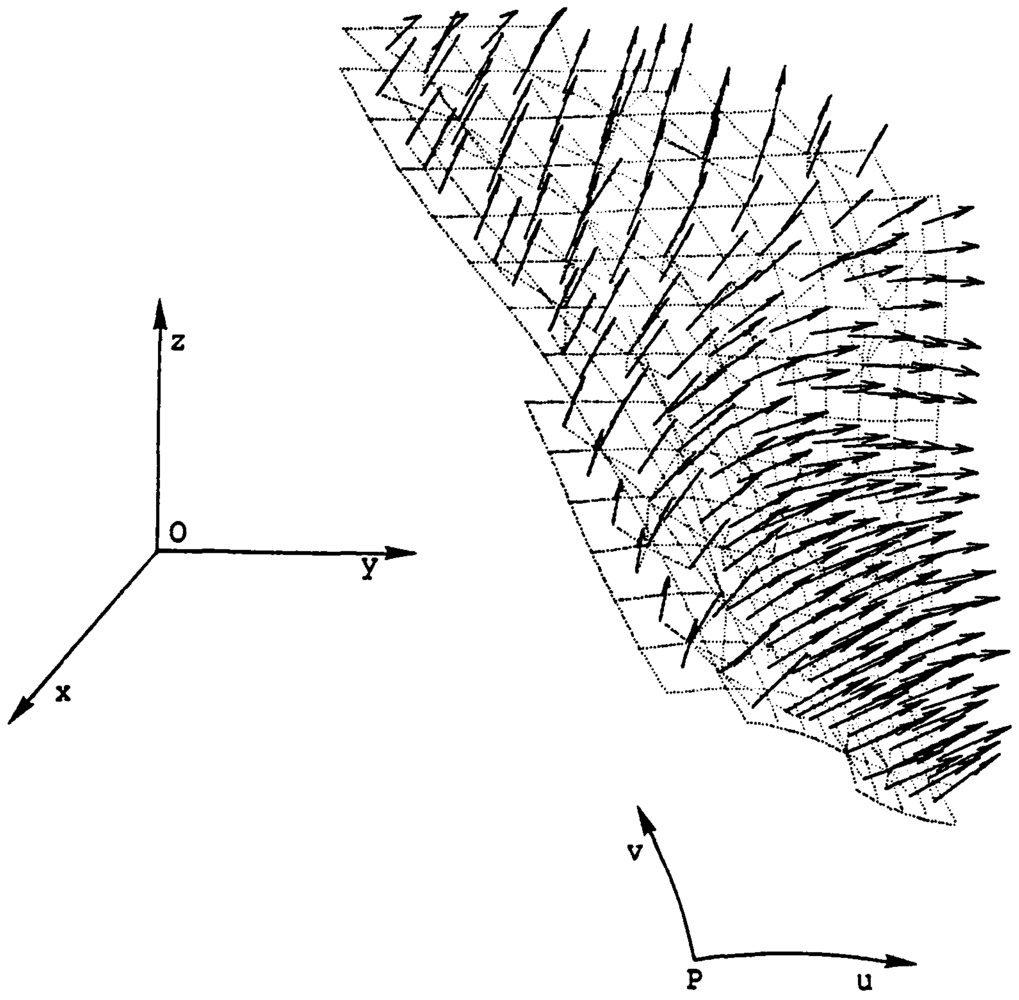


Figure 3.7 Quadtree of normal vectors (normal vectors solid, tree branches and patches dotted)

The next three examples use 3D surface models which were obtained from sources other than our surface acquisition method. The data for these surfaces had to be converted first into the standard surface representation. The first two examples demonstrate objects modeled by large numbers of sheets of bicubic patches which define closed 3D volumes; the third example represents open surface model formed by a single surface sheet. The first example is a surface description of an F100 engine blade designed by an external CAD/CAM system. The data for the 'F100' surface model contains:

47	sheets
3205	bounding volumes
2853	control points
2300	patches at levels 0
2 to 7	quadtree levels in a sheet

Four views of the blade are shown in Figure 3.8, drawn as line-drawings of the patch control points at level 1 of the quadtrees. Four synthetic images of the blade, using level 0 of the quadtrees, are shown in Figure 3.9. In the bottom two images the blade is placed on a planar surface and casts a shadow. The second example is a model of a J79 turbine blade. The data for the 'J79' model contains:

63	sheets
3819	bounding volumes
3551	control points
2519	patches at level 0
2 - 7	quadtree levels in a sheet

Four line drawings of this blade are shown in Figure 3.10. Four synthetic images of the blade, in orthographic pro-

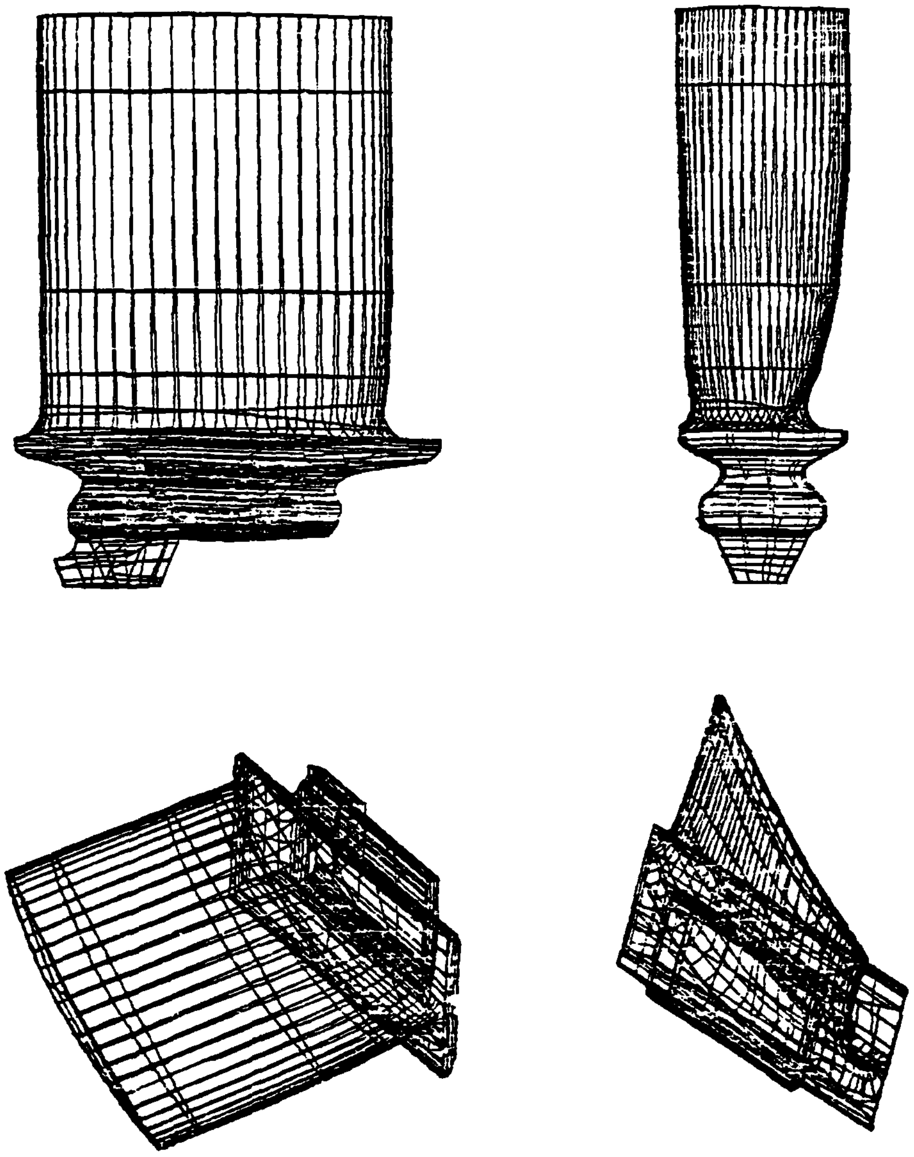


Figure 3.8 Four views of an F100 engine blade (3D network of lines)

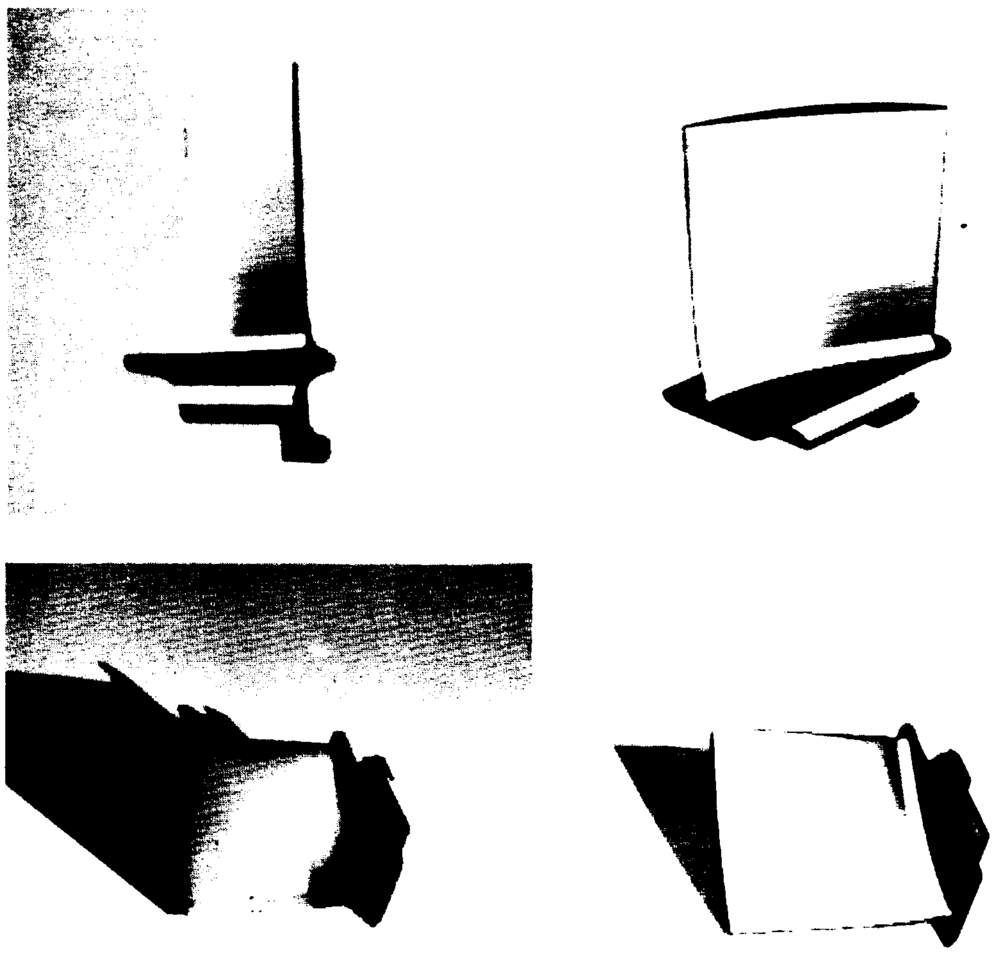


Figure 3.9 Four images of an F100 engine blade (3D quad-trees of bicubic patches)

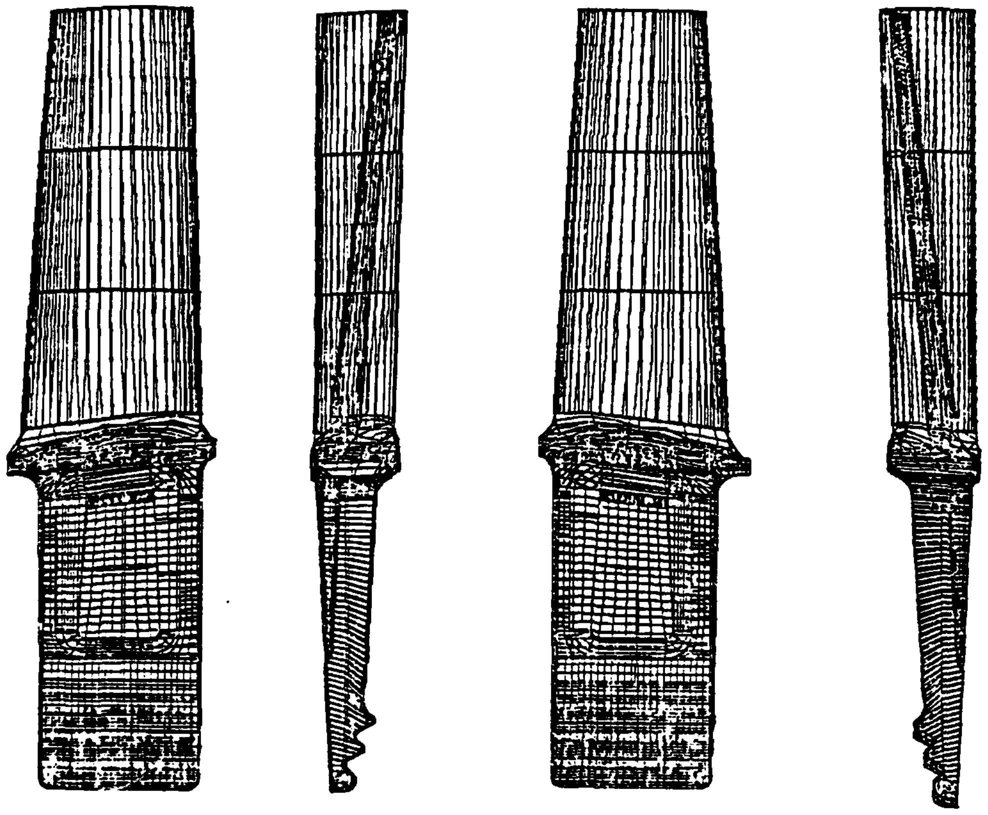


Figure 3.10 Four views of a J79 turbine blade (3D network of lines)

jection, are shown in Figure 3.11. The line drawings show quadtree data at level 1; the images show quadtree data at level 0. The two blade models shown in Figures 3.9 and 3.11 were assigned highly specular reflections to simulate a metalic surface.

The third example represents a 3D terrain model obtained from a digital contour map of the Fall River Pass Quadrangle in Colorado. The actual area of the modeled terrain is 6.6 x 8.5 miles. The ratio of the vertical scale to the horizontal scales in this model is 7:1:1 which significantly exaggerates the terrain's elevation. The surface model 'TERRAIN' contains the following data:

1	sheet
5489	bounding volumes
4218	control points
4088	patches at level 0
8	quadtree levels

There are four views of the terrain, drawn at level 2 of the quadtree, shown in Figure 3.12. There are four similar views of the terrain shown in the images of Figure 3.13. The first image (upper left) was generated from data at level 0 of the quadtree which contains 56 x 76 patches. An image of the original contour map is shown under the terrain, mapped on a rectangular plane. The other three images were generated from data at level 2 of the quadtree which contains 14 x 19 patches. Notice the difference in detail between the two representations. The terrain is shown from SSW (south-southwest), SSE, NNE, and NNW, respec-

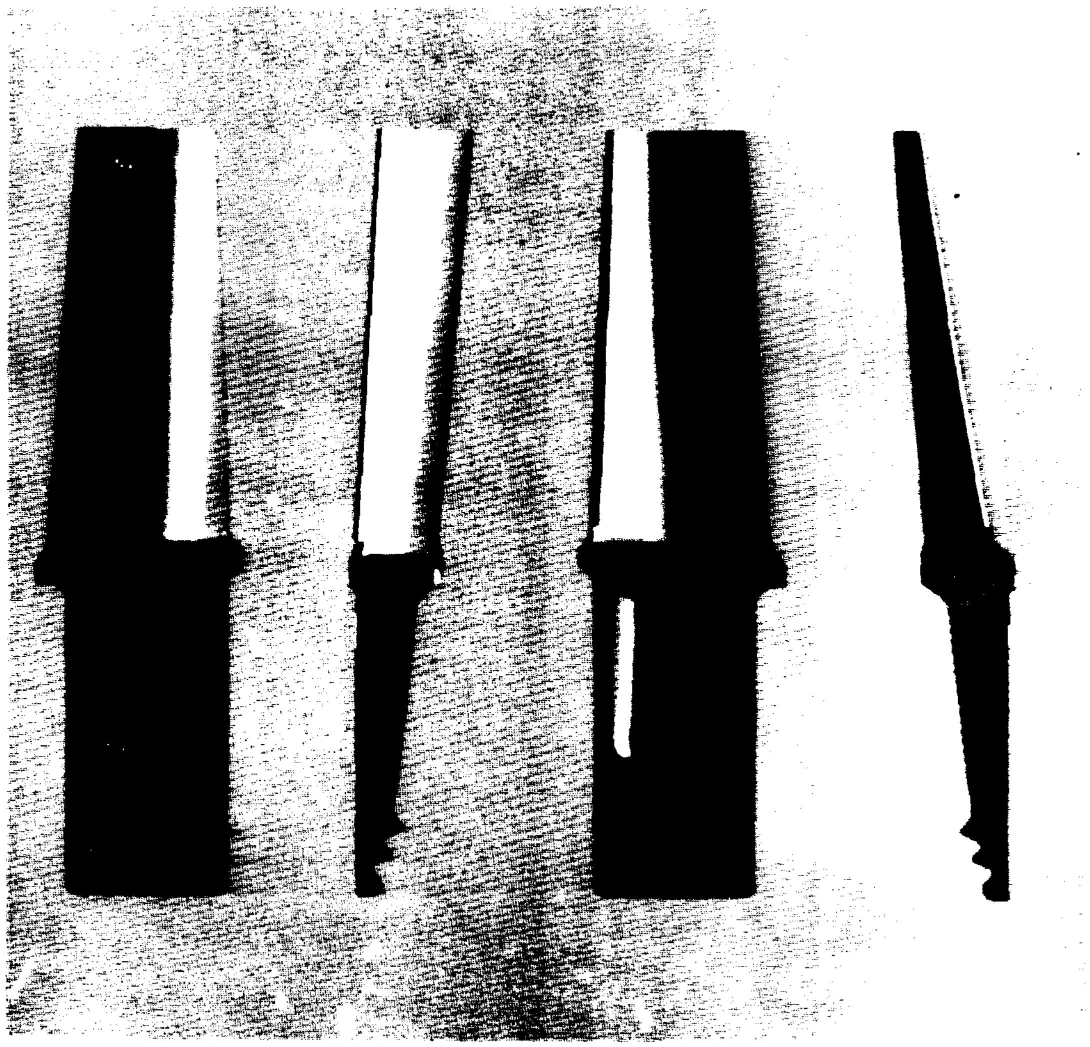


Figure 3.11 Four images of a J79 turbine blade (3D quad-trees of bicubic patches)

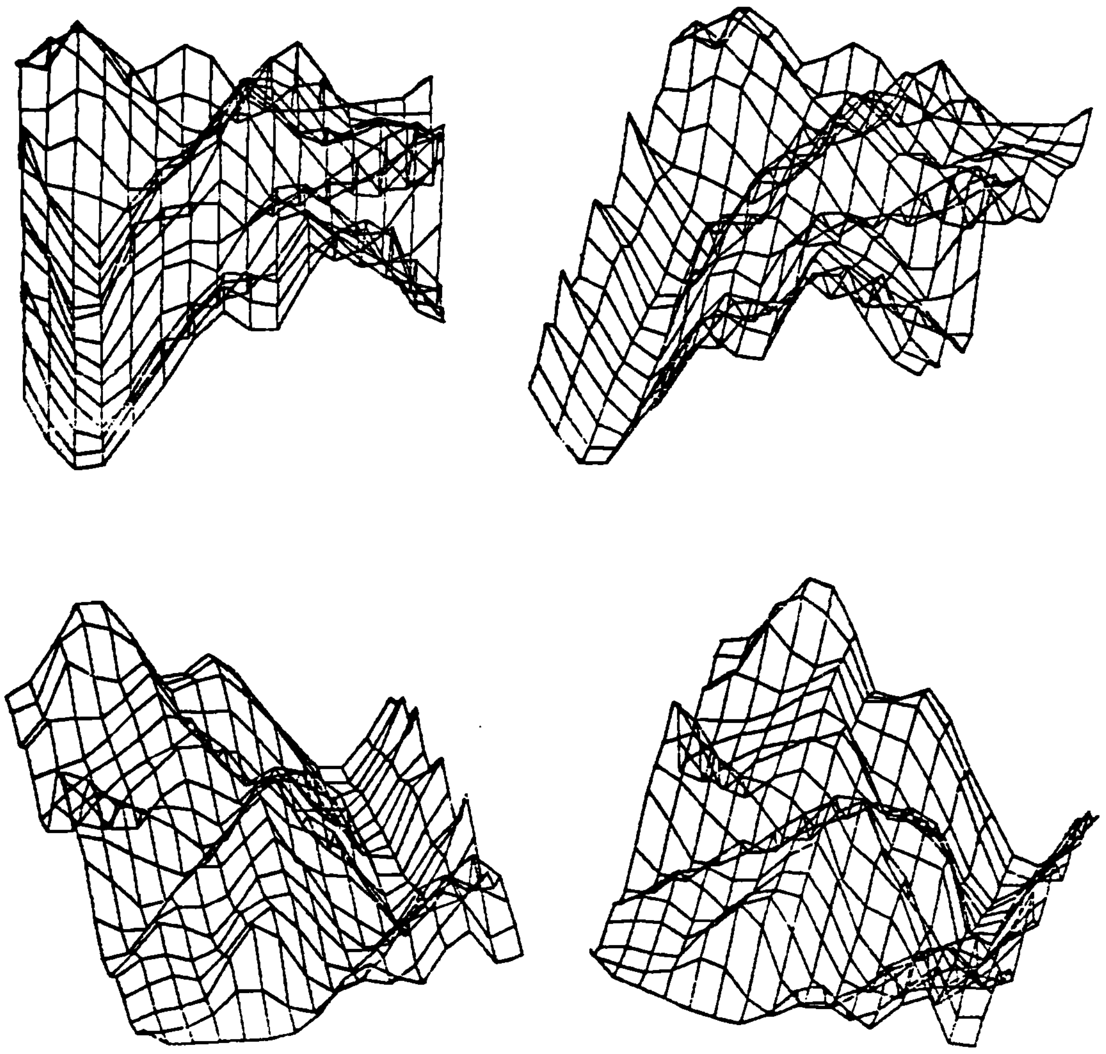


Figure 3.12 Four views of a terrain model (3D network of lines)

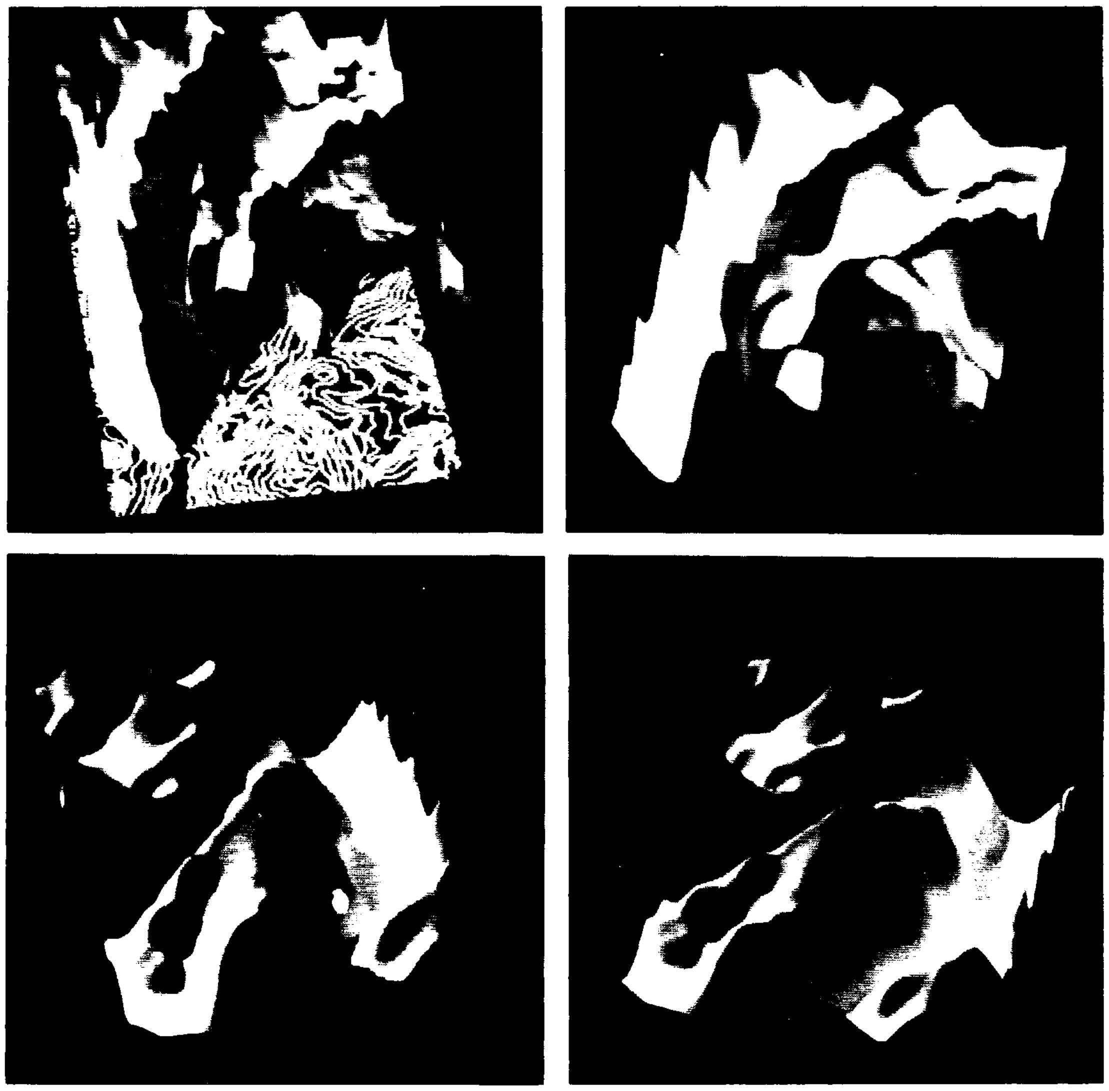


Figure 3.13 Four images of a terrain model (3D quadtree of bicubic patches)

tively, with the sun at SE, 30 degrees above the horizon. The colors of the surface are computed as a function of elevation from a look-up table; they range from dark green in the valleys through light green, yellow, orange, and light brown to dark brown at the mountain peaks.

The solid modeling capabilities of this system are demonstrated in the last two examples given in this chapter. First, the logical set operations are shown on two volume primitives [Figure 3.14]. Volume primitive 'A' is formed by union of three surface half-spaces: two spheres and one cylinder; similarly, primitive 'B' is formed by union of two planes and an ellipsoid [Figure 3.14(a)]. The union, difference, and intersection operations, applied to the two volume primitives, are shown in Figures 3.14(b), (c), and (d), respectively.

Finally, two section views of the J79 blade are shown in Figure 3.15. They are made by intersection and difference, respectively, of the blade model and an invisible parallelepiped which encloses the left half of the blade. One face of the parallelepiped intersects the blade in the vertical and front-back directions. Since the parallelepiped is invisible the sectioned blade appears hollow; the very dark surface is the back surface of the blade in shadow of the front surface.

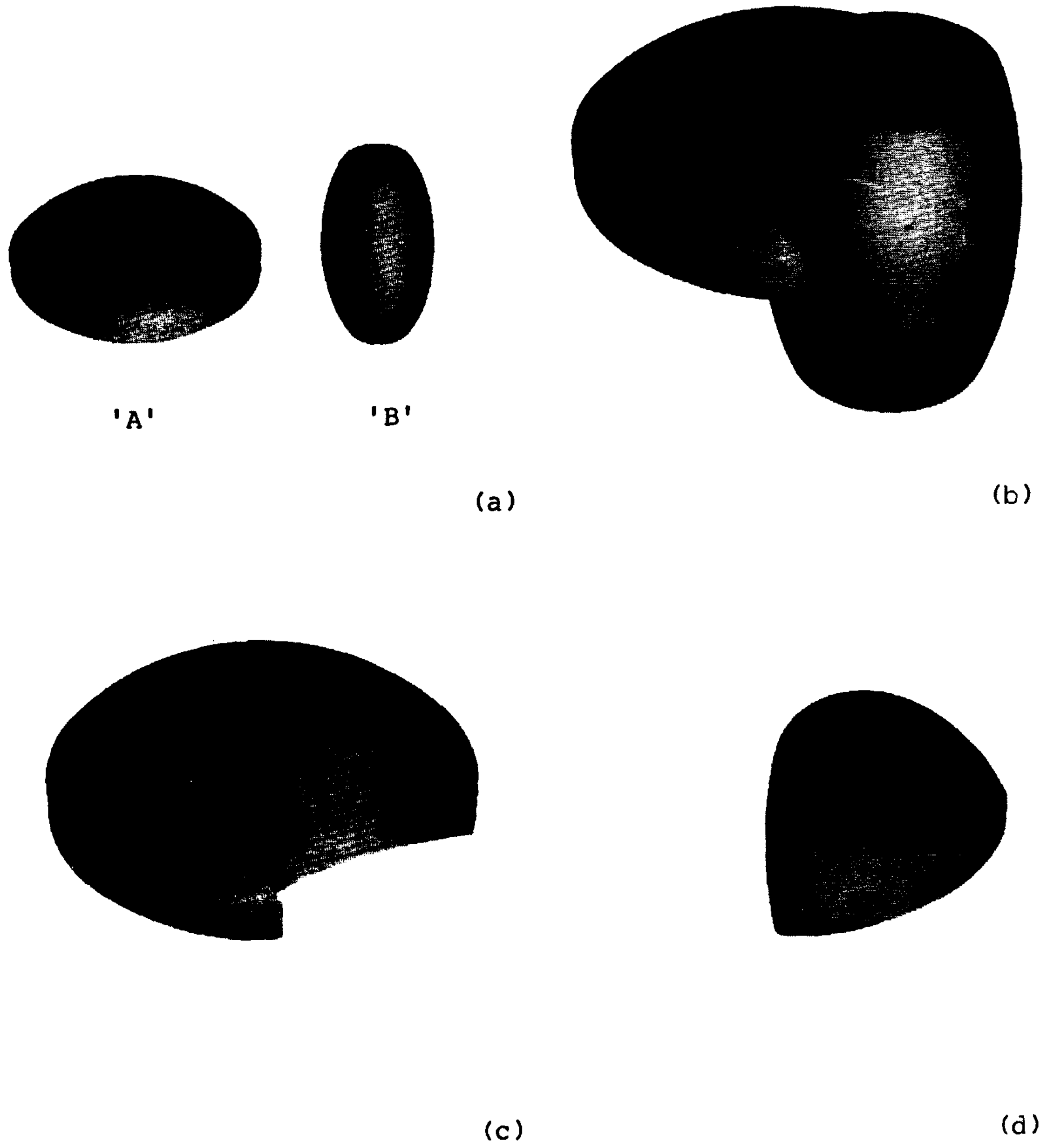
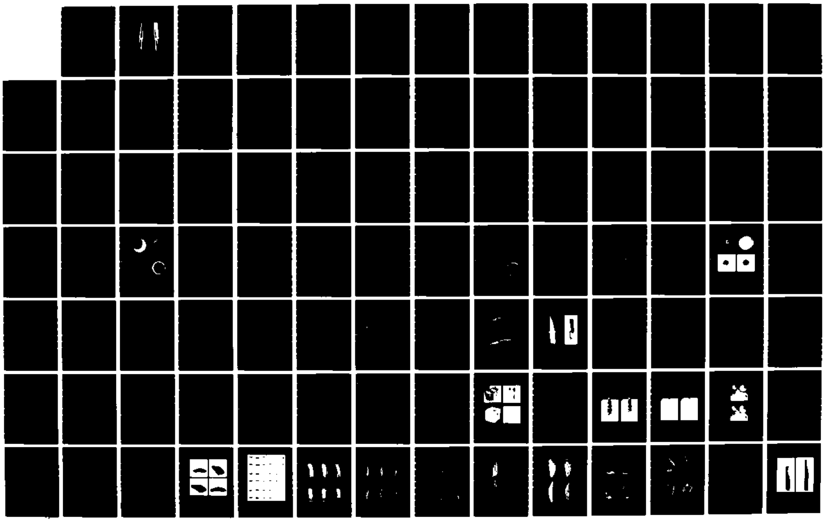
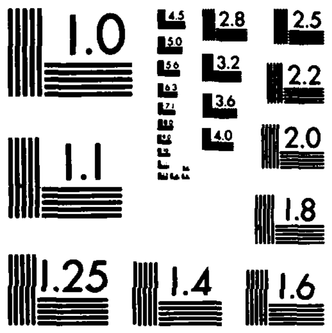


Figure 3.14 Modeling of solids: (a) two volume primitives, (b) union, (c) difference, (d) intersection

AD-A132 392 GENERATING THREE-DIMENSIONAL SURFACE MODELS OF SOLID OBJECTS FROM MULTIPL..(U) RENSSELAER POLYTECHNIC INST TROY NY IMAGE PROCESSING LAB M POTMESIL OCT 82 2/3

UNCLASSIFIED IPL-TR-033 N00014-82-K-0301 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

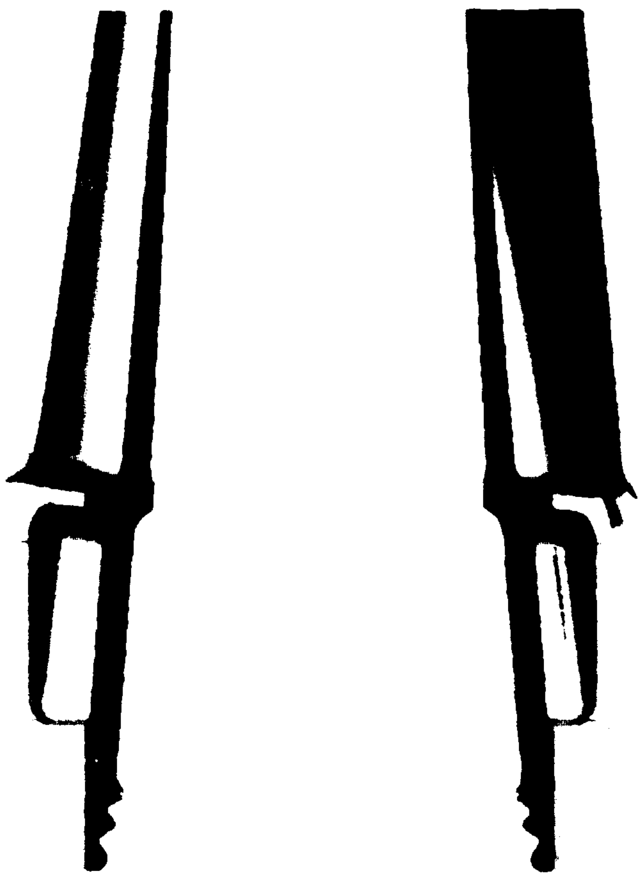


Figure 3.15 Two images of section views of the J79 turbine blade model made by intersection and difference of the blade and an invisible parallelepiped

3.8 Summary

In this chapter we presented an experimental modeling system for hierarchical representation of 3D surfaces, volumes, and objects. The system is intended for a wide range of applications, from CAD/CAM to computer vision. Desirable features such as good user interface for designing new surfaces and editing existing surfaces, or automatic testing for validity and consistency of representation (detection of nonsense objects) that are mandatory in an actual modeling system are beyond the scope of this work. All application algorithms which process this surface representation use a common ray-tracing traversal procedure to obtain a list of intersections of a ray with the surfaces in a directed, acyclic graph. Surface data is usually provided to the system as bivariate surface elements which are then structured into hierarchical quadtrees. However, representation by polyhedral and quadric surfaces is also available. New surface elements can be simply added to the system, provided that the five geometric operations (Section 3.5) are defined. These are the only operations in the entire system which depend on the type of surface representation.

CHAPTER 4

SURFACE MATCHING

Following the prerequisite processing described in the last two chapters, we finally reach the heart of this work - matching of 3D surfaces. In the context of this work, matching of 3D surfaces refers exclusively to finding a spatial registration of two 3D surface descriptions that maximizes their shape similarities. If a measurement of the shape similarities is acceptable then the surfaces are said to match, and the spatial registration aligns them. In order to find such a match, the two surface descriptions must, of course, completely or partially overlap, i.e., there must be a set of surface points common to both surface descriptions. Note that we do not refer here to other, possibly similar, types of matching such as (a) matching 2D projections with 2D models of 3D objects [17,29,14], (b) matching 2D projections with 3D models [9,11], or (c) relational matching of features of 3D objects and 3D models [69].

There are a number of problems where two or more 3D surfaces need to be matched. For example, a partial 3D surface description of an unknown object is obtained from one (stereo) vantage point. The object is to be recognized by matching this partial 3D surface description with a set of 3D models of various objects. Or pieces of broken pottery or fragments of objects need to be matched to allow reconstruction of the original objects. Or a 3D description of surface below an aircraft needs to be matched with a 3D terrain model to determine accurately the position of the aircraft. In the scope of this work, we need to match 3D surface segments of an object, which were generated while the object was in several stable positions and parametrized by several parameter systems, in order to build a complete 3D model of the object.

The matching algorithm of two 3D surfaces, which may completely or partially overlap, consists of two major steps: (1) initial estimates of the surface registration are computed by alignment of known points on both surfaces or alignment of surface-normal vectors representing surface orientations, and (2) a heuristic search improves these estimates by varying transformation parameters to find an acceptable solution. The measurement of shape similarity between the two surfaces is computed by an evaluation function from the following information, obtained for a number of points distributed on both surfaces: (1) Euclidean distance to the other surface, (2) angular difference between normal vectors, and (3) difference in local surface curvatures. The ray-tracing traversal of a surface description computes this information. Given a point on one surface and the surface-normal vector at this point it finds (1) the nearest intersection with the other surface, (2) the surface-normal vector at the point of intersection, and (3) the surface curvature at the point of intersection. The matching algorithm is independent of the surface representation which is confined only to the ray-tracing process.

Following the description of the matching problem, we present a number of algorithms which merge, into a single concise description, those overlapping surfaces that were either obtained in the same object coordinate system, or transformed into the same object coordinate system by the matching algorithm. The first algorithm merges two parametrized surfaces into a common 2D parameter coordinate system by a transformation of one parameter coordinate system. The second algorithm and its variations reparametrize the surface descriptions into a new parameter coordinate system by projecting this system on the surfaces. They can also be used to parametrize surfaces given only in algebraic representations. The third and last merging algorithm, which is applicable only to surface descriptions of a solid object, converts the surface representations into a polyhedral volume representation made of rectangular parallelepipeds. This solid representation can be further converted into the octree representation [44].

Finally, this chapter closes with descriptions of three applications of the matching algorithm to (1) generation of complete 3D models, (2) surface and object recognition, and (3) surface and volume segmentation with surface and volume primitives. The first application also uses the merging algorithms. It has been implemented and tested with several objects. The last two applications are described as suggested approaches to these problems.

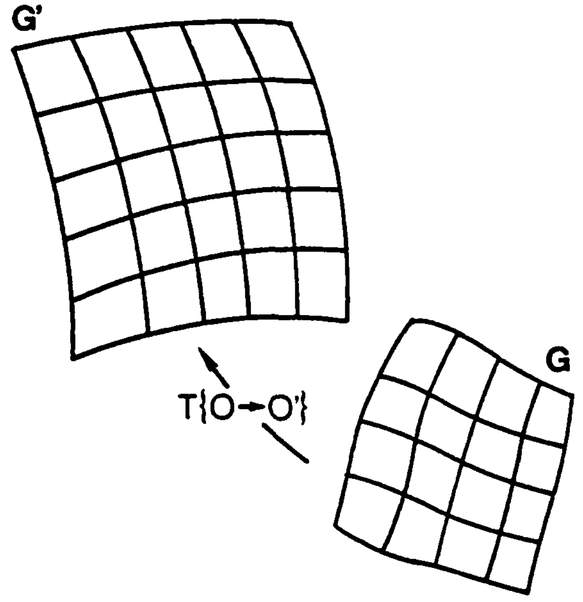
4.1 Types of Matching

There are two basic types of surface matching [Figure 4.1] that are of interest in this work. Given two 3D surface descriptions, represented by graphs G and G':

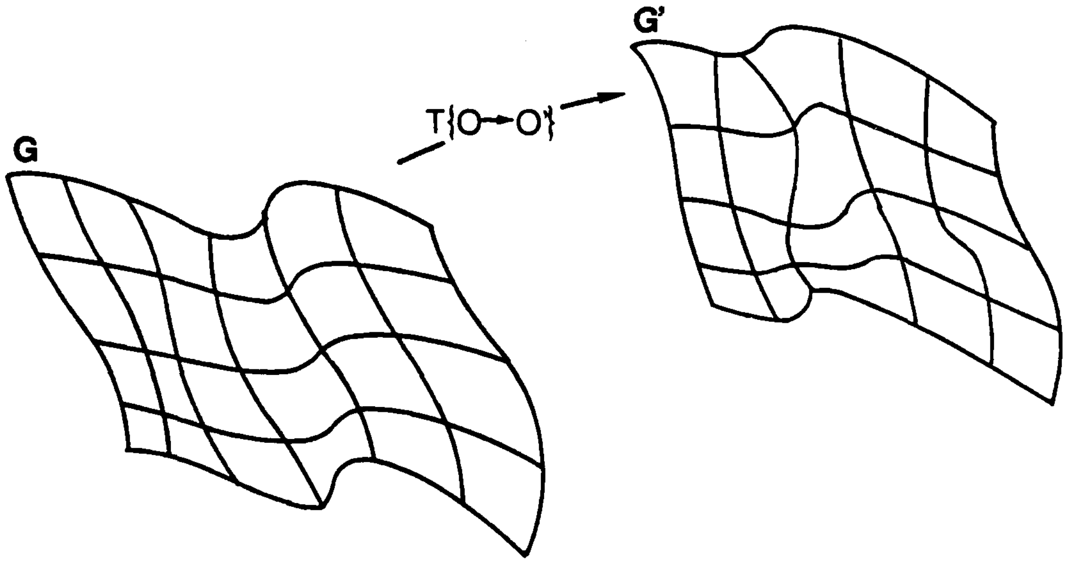
(a) surface description G is completely contained in description G' [Figure 4.1(a)]; that is, the intersection set of G and G' is G (e.g., a surface segment is being matched with a complete object model); and

(b) surface description G is only partially contained in description G' [Figure 4.1(b)]; that is, the intersection set of G and G' is a subset of G and a subset of G' (e.g., two overlapping surface segments of an object are being matched).

These two types of matching differ in the approach that the matching algorithm uses in generating the initial estimates of the surface registrations. In case (a), it emphasizes the spatial registrations that completely match G to G'. In case (b), it emphasizes the spatial registrations that match only a portion of G to a portion of G'. The first type of



(a)



(b)

Figure 4.1 Types of surface matching: (a) G completely overlaps G', (b) G partially overlaps G'

surface matching is used for object recognition. The second type is used for generation of complete object models, and for their segmentation with surface and volume primitives.

A similar type of surface matching could also be used for solving the 3D "jig-saw puzzle" problem where surface segments with only common boundary curves (e.g., broken pottery pieces) need to be matched and spatially registered. In this problem each surface segment would be extrapolated and labeled as either "actual" or "extrapolated" sections. The matching algorithm would then compute spatial registrations by matching the "actual" sections of G with the "extrapolated" sections of G', and the "extrapolated" sections of G with the "actual" sections of G'.

4.2 Surface Transformations

The goal of the matching algorithm is to compute a rigid 3D transformation that matches two surface descriptions and aligns them into the same object coordinate system. In this section we describe two types of rigid 3D transformations [Figure 4.2], used by the matching algorithm, and the computational methods that generate them. The two transformations consist of different numbers of transformation parameters. The first transformation is a general transformation between two arbitrarily-oriented object coordinate systems. It consists of three translation, three rotation,

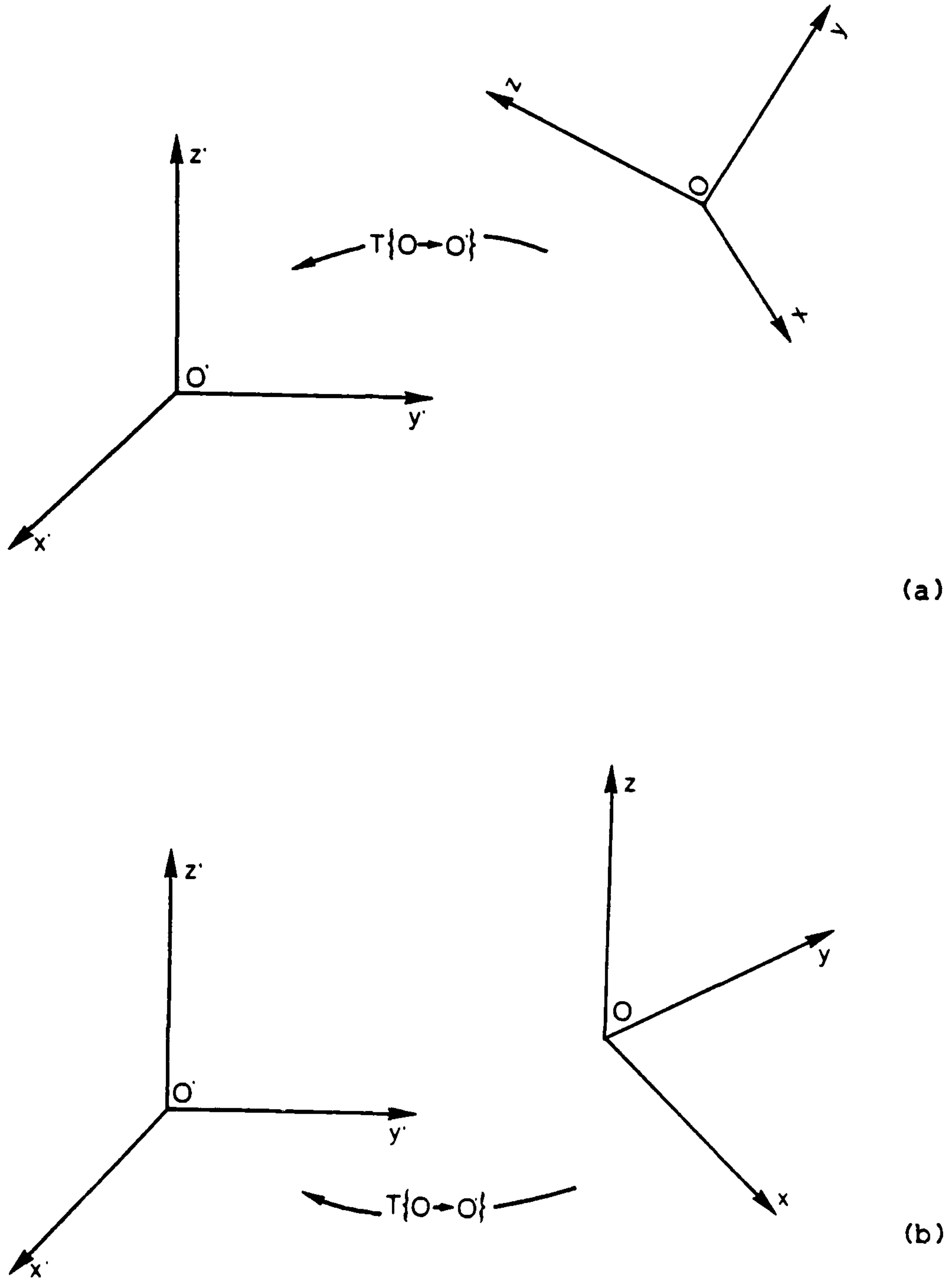


Figure 4.2 Surface transformations: (a) general, (b) limited to $z' = z$

and, optionally, three scale parameters. The second transformation is a limited transformation between two object coordinate systems which have one of the three axes parallel. It consists of two translation, one rotation, and, optionally, two scale parameters. It is used only when there is à priori knowledge that the object, whose surface segments are being matched, remained in the same stable position. The scale parameters are normally not used since it is assumed that all the surface descriptions have been obtained to the same scale - the actual size of the surfaces. They would only be used for object segmentation with surface and volume primitives defined at some standard size.

A general transformation $T\{O_k \rightarrow O_{k'}\}$ [Figure 4.2(a)] from homogeneous object coordinate system $O(x,y,z,w)_k$ to homogeneous object coordinate system $O(x,y,z,w)_{k'}$ is expressed as:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}_{k'} = T\{O_k \rightarrow O_{k'}\} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}_k = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}_k \tag{4.1}$$

This transformation consists of the following transformation parameters: three translations (tx, ty, tz), three rotations (rx, ry, rz), and, optionally, three scale factors (sx, sy, sz). The twelve unknown coefficients of $T\{O_k \rightarrow O_{k'}\}$

can be determined from the coordinates of four non-coplanar points in $O(x,y,z,w)_k$ and their corresponding images in $O(x,y,z,w)_{k'}$.

A limited 3D transformation $T\{O_k \rightarrow O_{k'}\}$ [Figure 4.2(b)] from homogeneous object coordinate system $O(x,y,z,w)_k$ to homogeneous object coordinate system $O(x,y,z,w)_{k'}$ with $z_k = z_{k'}$ is expressed as:

(4.

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}_{k'} = T\{O_k \rightarrow O_{k'}\} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}_k = \begin{bmatrix} t_{11} & t_{12} & 0 & t_{14} \\ t_{21} & t_{22} & 0 & t_{24} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}_k$$

This transformation consists of these transformation parameters: two translations (tx, ty), one rotation (rz), and, optionally, two scale factors (sx, sy). The six unknown coefficients of this $T\{O_k \rightarrow O_{k'}\}$ can be determined from the $\underline{x}$ and $\underline{y}$ coordinates of three non-colinear points in $O(x,y,z,w)_k$ and their corresponding images in $O(x,y,z,w)_{k'}$. This transformation is available to simplify the matching process when the two surface segments being matched have been obtained from an object in the same stable position with respect to the x-y planes in the $O(x,y,z,w)_k$ and $O(x,y,z,w)_{k'}$ coordinate systems.

In general, a rigid 3D transformation, composed of translation and rotation, is defined as:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_{14} \\ t_{24} \\ t_{34} \end{bmatrix} \qquad (4.3)$$

or, for short,

$$\bar{h} = \bar{R}\,\bar{h} + \bar{t} \qquad (4.4)$$

where $\bar{t}$ is a translation (displacement) transformation and $\bar{R}$ is an <u>orthogonal</u> rotation transformation which imposes the condition:

$$\bar{R}\,\bar{e} \cdot \bar{R}\,\bar{f} = \bar{e} \cdot \bar{f} \qquad (4.5)$$

for all vectors $\bar{e}$ and $\bar{f}$ in O(x,y,z,w). This condition preserves the distance between any two points and the angle between any two vectors. In addition, the determinant of $\bar{R}$ must be positive to preserve orientation (i.e. to avoid reflections) in the right-handed object coordinate system O(x,y,z,w). The inverse, $\bar{R}^{-1}$, of an orthogonal matrix $\bar{R}$ is equal to its transpose $\bar{R}^t$. Note that $\bar{R}$ is the upper-left 3 x 3 principal submatrix of $\bar{T}$ in equation (4.1), and $\bar{t}$ is the upper-right 3 x 1 submatrix of $\bar{T}$. A transformation which contains scale factors other than unity is not a true rigid transformation because it does not preserve distances but only angles.

The matching algorithm uses two methods to compute the object transformation matrix (4.1) (the limited transformation (4.2) is only a special case of (4.1)). The first method composes a transformation matrix for each transformation parameter and multiplies these matrices into the total transformation [Appendix C.1.2]. A rotation matrix composed by this method is guaranteed to be always orthogonal.

The second method determines the transformation from matched points in the two object coordinate systems. Given a point $\overline{h}(x,y,z,w)$ in $O(x,y,z,w)_k$ and its image $\overline{h}'(x,y,z,w)$ in $O(x,y,z,w)_{k'}$ we rewrite (4.1) as a system of three linear equations with twelve unknowns:

$$\begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x & y & z & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & y & z & 1 \end{bmatrix}_k \begin{bmatrix} t_{11} \\ t_{12} \\ \cdots \\ \cdots \\ t_{33} \\ t_{34} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{k'} \tag{4.6}$$

or, for short,

$$\overline{D}\ \overline{T} = \overline{D}' \tag{4.7}$$

With three additional points we can express system (4.6) as twelve equations with twelve unknown coefficients of $\overline{T}$. In general, for $\underline{N} \geq 4$ points the least-squares method is used to minimize the sum of squares of distances between the matched pairs of points $\overline{h}_i$ and $\overline{h}'_i$ in equation (4.4):

$$\min\left[\sum_{i=1}^{N} \left| \overline{h}'_i - \overline{R}\ \overline{h}_i + \overline{t}\ \right|^2\right] \tag{4.8}$$

which yields the standard least-squares solution:

$$\begin{bmatrix} \overline{D}_1 \\ \cdot\cdot \\ \overline{D}_N \end{bmatrix}^t \begin{bmatrix} \overline{D}_1 \\ \cdot\cdot \\ \overline{D}_N \end{bmatrix} \overline{T} = \begin{bmatrix} \overline{D}_1 \\ \cdot\cdot \\ \overline{D}_N \end{bmatrix}^t \begin{bmatrix} \overline{D}'_1 \\ \cdots \\ \overline{D}'_N \end{bmatrix} \tag{4.9}$$

This solution, however, does not guarantee $\overline{R}$ to be orthogonal. Once we have obtained the translation vector $\overline{t}$ from (4.9), we can improve the othogonality of $\overline{R}$ by also using the inverse transformation of (4.4), namely:

$$\overline{h} = \overline{R}^{-1}(\overline{h}'-\overline{t}) = \overline{R}^t(\overline{h}'-\overline{t}) \tag{4.10}$$

to minimize the squares of distances, as in (4.8), with the transformation $\overline{R}$ and its inverse $\overline{R}^t$:

$$\min\left[\sum_{i=1}^{N} \mid \overline{h}'_i-\overline{R}\ \overline{h}_i+\overline{t}\ \mid^2 + \mid \overline{R}^t(\overline{h}'_i-\overline{t})-\overline{h}_i\ \mid^2\right] \tag{4.11}$$

The solution to (4.11) is similar to (4.9), and still not necessarily orthogonal. Note that to obtain an exactly orthogonal $\overline{R}$ we would have to solve a system of nine simultaneous quadratic equations. To avoid this, the matrix $\overline{R}$ obtained from (4.11) is converted to an orthogonal matrix with the following procedure:

(4.12)

(1) convert each row (column) vector, $\overline{r}_i$, of $\overline{R}$ to a unit vector;

(2) find two most orthogonal row (column) vectors, $\overline{r}_i$ and $\overline{r}_j$:

$\mid \overline{r}_i \cdot \overline{r}_j \mid < \mid \overline{r}_i \cdot \overline{r}_k \mid$, and

$\mid \overline{r}_i \cdot \overline{r}_j \mid < \mid \overline{r}_j \cdot \overline{r}_k \mid$;

(3) make $\overline{r}_i$ and $\overline{r}_j$ orthogonal:

$\overline{r}_i = \overline{r}_i - (\overline{r}_i \cdot \overline{r}_j)\ \overline{r}_j$;

(4) convert new $\overline{r}_i$ to a unit vector;

(5) make $\overline{r}_k$ othogonal to $\overline{r}_i$ and $\overline{r}_j$:

$\overline{r}_k = \overline{r}_i \times \overline{r}_j$;

(6) preserve orientation:

if $|\overline{R}| < 0$ then $\overline{r}_k = -\overline{r}_k$.

In summary, a rigid 3D transformation is computed from the coordinates of N points in O(x,y,z,w) and their images in O'(x,y,z,w) by equation (4.8) which obtains translation $\overline{t}$, equation (4.11) which obtains rotation $\overline{R}$, and procedure (4.12) which makes $\overline{R}$ orthogonal. The last step prevents skewed and sheared transformations.

4.3 Matching Algorithm

The matching algorithm computes a 3D rigid transformation $\overline{T}$, composed of a rotation matrix $\overline{R}$ and a translation vector $\overline{t}$, which spatially registers (aligns) surface description G with surface description G'. This transformation occurs when the orientation and shape differences between the two surfaces are minimized (i.e., the shape similarity is maximized). The orientation and shape differences are evaluated at a number of evaluation points on surface G from information provided by the ray-tracing procedure

[Chapter 3]. For a ray in the direction of the surface-normal vector at an evaluation point, $\overline{h}(x,y,z,w)_i$, on surface $\underline{G}$ this procedure finds the nearest intersection with $\underline{G}'$, the surface-normal vector, and the normal surface curvature, both evaluated at the point of intersection. The differences in surface orientation and shape are computed from these components:

(1) <u>position difference</u>, $p_i(\overline{T})$, is the 3D Euclidean distance in the object coordinate system $O'(x,y,z,w)$:

$$p_i(\overline{T}) = \left| \overline{T}\, \overline{h}(x,y,z,w)_i - \overline{h}'(x,y,z,w)_i \right| \tag{4.13}$$

where

$\overline{T}$	= current transformation from $O(x,y,z,w)$ to $O'(x,y,z,w)$
$\overline{h}(x,y,z,w)_i$	= an evaluation point on surface $\underline{G}$
$\overline{h}'(x,y,z,w)_i$	= the point of intersection on surface $\underline{G}'$

(2) <u>orientation difference</u>, $a_i(\overline{T})$, is the angular difference of the surface unit-normal vectors oriented to the "outside" of the surfaces:

$$a_i(\overline{T}) = \left| (\overline{R}\, \overline{N}_i) \cdot \overline{N}'_i - 1.0 \right| \tag{4.14}$$

where

$\overline{R}$	= current rotation from $O(x,y,z,w)$ to $O'(x,y,z,w)$
$\overline{N}_i$	= unit normal vector at $\overline{h}(x,y,z,w)_i$
$\overline{N}'_i$	= unit normal vector at $\overline{h}'(x,y,z,w)_i$

(3) curvature difference, $c_i(\overline{T})$, is the magnitude of difference of the surface normal curvatures:

$$c_i(\overline{T}) = | q_i - q'_i | \tag{4.15}$$

where

q_i = normal surface curvature at $\overline{h}(x,y,z,w)_i$

q'_i = normal surface curvature at $\overline{h}'(x,y,z,w)_i$

The total surface difference, $d_i(\overline{T})$, at this point is then defined as a linear combination of these three components:

$$d_i(\overline{T}) = w_p p_i(\overline{T}) + w_a a_i(\overline{T}) + w_c c_i(\overline{T}) \tag{4.16}$$

where

w_p = a weight of position distance $p_i(\overline{T})$

w_a = a weight of angular difference $a_i(\overline{T})$

w_c = a weight of curvature difference $c_i(\overline{T})$

Given N evaluation points on surface G, which are used in computing the surface registration, we seek a 3D transformation $\overline{T}$ such that:

$$D(\overline{T}) = \sum_{i=1}^{N} w_i d_i(\overline{T}) < \text{epsilon} \tag{4.17}$$

where

$d_i(\overline{T})$ = surface difference at point $\overline{h}(x,y,z,w)_i$

w_i = a weight of point $\overline{h}(x,y,z,w)_i$

epsilon = an acceptable difference between the two surface descriptions G and G' for a match

An additional condition requires that a correct surface match be found for at least $\underline{M}$ of the $\underline{N}$ evaluation points where $4 \ll \underline{M} \leq \underline{N}$. Notice that a ray from $\overline{h}(x,y,z,w)_i$ may (a) miss $\underline{G}'$, in which case $d_i = 0$, or (b) find an incorrect match with $\underline{G}'$ [Figure 4.3], in which case:

(4.18)

$$|\overline{h}'(x,y,z,w)_i - \overline{h}'(x,y,z,w)_j| \gg |\overline{h}(x,y,z,w)_i - \overline{h}(x,y,z,w)_j|$$

rather than

(4.19)

$$|\overline{h}'(x,y,z,w)_i - \overline{h}'(x,y,z,w)_j| \cong |\overline{h}(x,y,z,w)_i - \overline{h}(x,y,z,w)_j|$$

for any point $\overline{h}(x,y,z,w)_j$, near $\overline{h}(x,y,z,w)_i$, and its intersection $\overline{h}'(x,y,z,w)_j$ on $\underline{G}'$ (i.e., as stated in Section 4.2: the distance of any two points must be preserved by $\overline{T}$). If the relationship (4.18) is valid for the point $\overline{h}(x,y,z,w)_i$ then $d_i = 0$.

Having defined the method which evaluates the surface difference, we shall now describe the actual search algorithm for a spatial registration which minimizes this difference between the two surfaces. A basic matching algorithm which would blindly wander in its task without any guidance can be outlined as:

(4.20)

A naive algorithm

(1) Select a set of $\underline{N}$ evaluation points on $\underline{G}$.

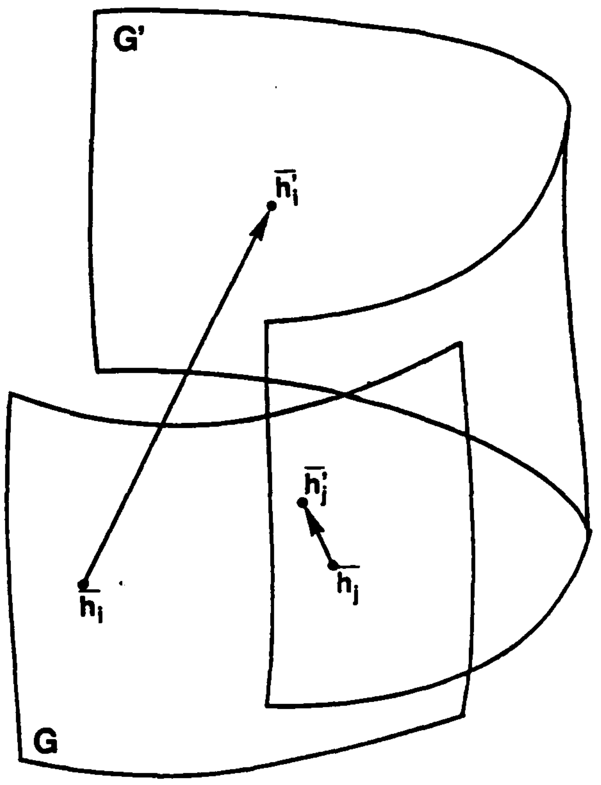


Figure 4.3 Correct ($\bar{h}_j$) and incorrect ($\bar{h}_i$) matches of evaluation points on surfaces $\underline{G}$ and $\underline{G}'$

(2) Generate a new transformation $\overline{T}$ of $\underline{G}$.

(3) Compute surface difference, d_i, at each evaluation point, $\overline{h}(x,y,z,w)_i$, transformed by $\overline{T}$, with $\underline{G}'$.

(4) If the total difference $D(\overline{T})$ is less than epsilon and $\underline{M}$ is much greater than 4 then the surfaces match and $\overline{T}$ aligns them, otherwise go to (2).

Clearly, an exhaustive search for the transformation parameters of $\overline{T}$ would be computationally prohibitive. A hill-climbing method, that would adjust each parameter in turn, would also be computationally excessive and probably would fail in a number of cases. A viable method to reduce the search task is to use a state-space search method [48, 49] with an evaluation function to guide the search for the transformation parameters of $\overline{T}$ that satisfy (4.17). This method represents the search process by a graph whose nodes are generated by successor operators which attempt to improve the node's surface registration. An evaluation function, $\underline{e}$, provides a ranking of the graph nodes to determine which nodes are most likely to be on a path to the solution and should, therefore, be expanded. An outline of this algorithm, adapted to the domain of our problem, can be specified as follows:

(4.21)

An ordered-search algorithm

(1) Generate start nodes, $\underline{n}_i$, and their $\overline{T}$ transforma-

tions; put them on list OPEN, compute $e(n_i)$ for each start node $\underline{n}_i$.

(2) If OPEN is empty exit with failure.

(3) Remove from OPEN a node $\underline{n}$ with the smallest $e(n)$ and put it on list CLOSED. If CLOSED becomes full exit with failure.

(4) If $D(\overline{T})$ of $\underline{n}$ is less than epsilon and $\underline{M}$ of $\underline{n}$ is much greater than 4, exit with solution $\overline{T}$ of $\underline{n}$.

(5) Expand node $\underline{n}$, generate all of its successors by computing their $\overline{T}$ transformations; for each successor, $\underline{n}_i$, compute $e(n_i)$.

(6) Put all successors, which are not already on OPEN or CLOSED, on OPEN and link them to $\underline{n}$. If OPEN becomes full exit with failure.

(7) For all successors which are already on OPEN or CLOSED, if new $e(n_i)$ is lower than the old $e(n_i)$ then replace it; move from CLOSED to OPEN all nodes whose value $e(n_i)$ was lowered. If OPEN becomes full exit with failure.

(8) Go to (2).

Each start node is the root node of a search tree; all the terminal nodes of a search tree are on list OPEN. The critical parts of this algorithm are the choice of the evaluation function $e(n)$, the test for the presence of a successor node on OPEN or CLOSED in steps (6) and (7), the selection of the evaluation points on surface $\underline{G}$ to be matched with $\underline{G}'$, and the strategies of the successor operators which (a) generate start (root) nodes of the search trees in step (1), and (b) expand (terminal) nodes on OPEN in step (5). These parts of the matching algorithm shall now be described in detail.

The evaluation function critically affects the search process. A function which is too generous will cause the expansion of too many nodes. On the other hand, a function which ignores the potential of some nodes can lead to a futile search. The currently used evaluation function, e(n), is a weighted sum of the surface difference $D(\overline{T})$, the inverse of the number of matched evaluation points, and the length of the tree path from the current node to its start node:

$$e(n) = w_D D(\overline{T}) + w_M/M + w_L L(n) \tag{4.22}$$

where

$D(\overline{T})$ = difference of $\underline{G}$, transformed by $\overline{T}$, and $\underline{G}'$

w_D = a weight of surface difference $D(\overline{T})$

M = number of matched evaluation points

w_M = a weight of inverse of number of matched points

$L(n)$ = the path lenght from $\underline{n}$ to its start node

w_L = a weight of path lenght $L(n)$

The first two terms of the evaluation function (4.22) provide <u>heuristic</u> information about the quality of the spatial registration.

The ordered-search algorithm has to check, in steps (6) and (7), whether a successor node is already present on OPEN or CLOSED. Each node contains the coordinates of a standard 3D vector, $\bar{H}$, transformed by the node's $\bar{T}$ transformation. If the distance of the transformed vector, $\bar{T}\,\bar{H}$, of a successor node is less than a selected tolerance from $\bar{T}\,\bar{H}$ of a node already on OPEN or CLOSED, then the transformation of the successor node is assumed to be the same as that of the node on OPEN or CLOSED and the successor node is already present there.

The set of evaluation points on surface $\underline{G}$, where the surface difference (4.17) is computed, should fairly represent the shape of the surface. There are two possible approaches, considered here, to the selection of these evaluation points: (a) find "critical" points at high surface curvature (e.g., surface vertices and edges), or (b) generate regularly spaced points on a parametric grid. The first approach, although potentially more powerful, suffers from a number of problems. It is difficult to find individual surface points with significantly higher curvature on relatively smooth, multiply-curved surface segments (see 'SURFACE.3' in Figures 2.9-11). On surfaces with constant curvature (e.g., spheres and cylinders) this method fails completely. On highly-curved surfaces these points tend to cluster along surface edges and be colinear. A search to locate points with maximum surface curvature is also computationally demanding.

In the second approach, adapted by the matching algorithm, the evaluation points are regularly spaced on a surface and hierarchically structured. A natural representation for this approach is the surface quadtree developed in Section 3.6. The control (corner) points of parametric patches are used as the evaluation points. In the initial stages of a search, the algorithm uses the control points from the high levels of a quadtree $\underline{G}$; as the value of $D(\overline{T})$ decreases, the control points are replaced by points from lower levels of the quadtree and their number, therefore, increases. Similarly, initially in a search the algorithm intersects rays from these evaluation points with patches stored in the high levels of quadtree $\underline{G}'$. As the value of $D(\overline{T})$ descreases, patches from lower levels are intersected. In summary, the matching algorithm, therefore, uses the hierarchical surface description as follows: while the estimate of the spatial registration is coarse, only a coarse surface description is used to evaluate $D(\overline{T})$; as the spatial registration improves, $D(\overline{T})$ is evaluated at more points with more accurate surface description. Selection of the quadtree level used in the evaluation of the surface difference is a function of $D(\overline{T})$:

$$\text{level} = \begin{cases} 0 & \text{if } D(\overline{T}) \leq D_{min}(1) \\ i & \text{if } D_{min}(i) < D(\overline{T}) \leq D_{max}(i) \\ \text{level}_{max} & \text{if } D_{max}(\text{level}_{max}-1) < D(\overline{T}) \end{cases} \quad (4.23)$$

where $0 < i < \text{level}_{max}$, and level_{max} is the highest level of a quadtree containing a surface approximation. An evaluation point may be assigned a weight proportional to its surface curvature. This weight is used in (a) evaluation of $D(\overline{T})$, and (b) generation of new $\overline{T}$ when a graph node is being expanded. For surface and volume primitives it may be desirable to define the evaluation points while creating a primitive.

There are two successor operators which generate the graph nodes of a search process. A start operator generates the start nodes of this process from the initial orientation of $\underline{G}$ in O(x,y,z,w). These nodes are put on OPEN in step (1) of algorithm (4.21). An expansion operator generates the successor nodes of the node with the currently best spatial registration in step (5) of this algorithm.

The start operator can use one of two approaches in generating the initial estimates of the surface match: (1) align known surface points, or (2) align orientation of surface-normal vectors. In the first method, there are $\underline{L}$ (≥ 4) points located on each surface. If they are not individually matched, all of their permutations must be matched, generating L! initial nodes. This number, however, can be substancially reduced if there is (a) à priori knowledge of their matches, (b) the matches can be generated with a method such as stochastic labeling [11]. Once the matches of individual points have been established, the transformation $\overline{T}$ is computed from equations (4.6-11) and procedure (4.12) [Section 4.2]. This initial estimate of the spatial registration $\overline{T}$, however, contains errors caused by measurements of the matched points, uncertainties of their matches, and the orthogonality requirement of the rotation matrix $\overline{R}$. This registration is, therefore, improved by a further search for a better alignment.

The second method generates the initial estimates of the spatial registration by alignment of surface-normal vectors present in the bounding volumes of the surface quadtrees and representing orientation of the surface elements within the volume [Section 3.3]. The two quadtrees, $\underline{G}$ and $\underline{G}'$, are traversed from their root nodes to bounding-volume nodes of selected size (volume), and the normal vectors in these volumes are spatially aligned. Since the four successor nodes in a quadtree are spatially ordered [Figure 3.4], the two strategies of matching completely or partially overlapping surfaces [Section 4.2] can be implemented. If there is à priori knowledge that the surfaces partially overlap, then each quadtree is traversed only to the bounding-volume nodes near the parametric boundaries of the surface description. However, if the surfaces completely overlap then each quadtree is traversed to all the bounding-volume nodes of the selected size. Each visited bounding-volume node in G is matched with each visited bounding-volume node in G'.

The surface-normal vectors in two bounding volumes, B and B', of G and G', respectively, are aligned as follows: the translation, $\overline{t}$, is computed by translating the center of B to the center of B':

$$\overline{t} = \begin{bmatrix} x_c' - x_c \\ y_c' - y_c \\ z_c' - z_c \end{bmatrix}, \qquad (4.24)$$

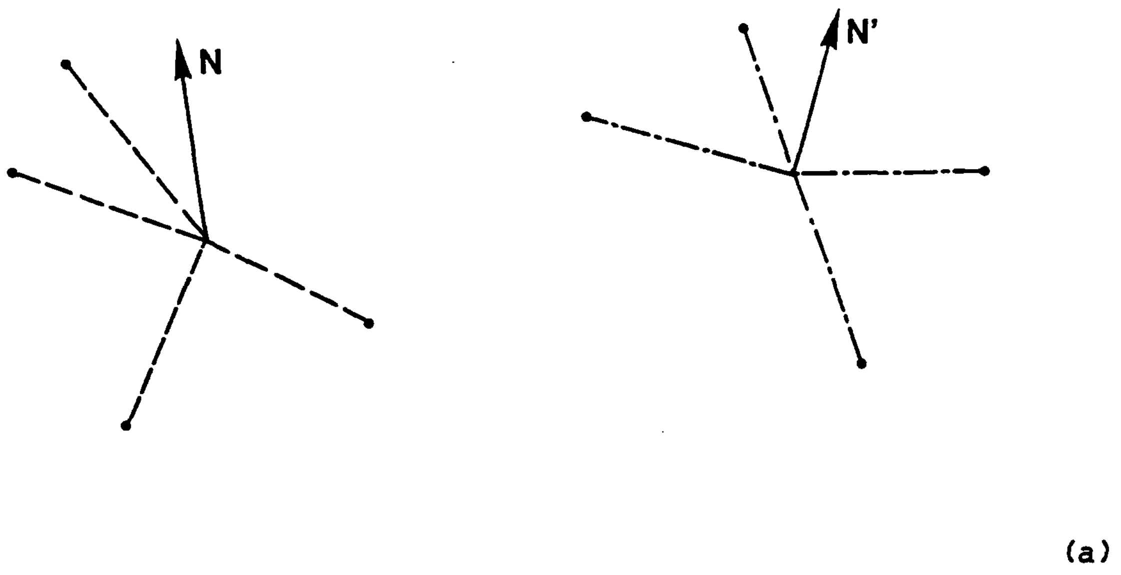
and the rotation, $\overline{R}$, of G around this point is computed by rotating $\overline{N}$ to $\overline{N}'$, and two additional orthogonal vectors $\overline{N}_1$ and $\overline{N}_2$ in G to $\overline{N}_1'$ and $\overline{N}_2'$ in G', so that:

$$\begin{aligned} \overline{N}' &= \overline{R}\,\overline{N} \\ \overline{N}_1' &= \overline{R}\,\overline{N}_1 \qquad (4.25) \\ \overline{N}_2' &= \overline{R}\,\overline{N}_2 \end{aligned}$$

where

$$\overline{N}\cdot\overline{N}_1 = \overline{N}\cdot\overline{N}_2 = \overline{N}_1\cdot\overline{N}_2 = 0$$
$$\overline{N}'\cdot\overline{N}_1' = \overline{N}'\cdot\overline{N}_2' = \overline{N}_1'\cdot\overline{N}_2' = 0.$$

These three pairs of orthogonal unit vectors are required to compute the nine coefficients of $\overline{R}$. After $\overline{N}$ has been rotated to $\overline{N}'$, surface G can still be rotated around $\overline{N}$ [Figure 4.4]. This rotation is determined by projecting the centers of bounding volumes which are the successors of B



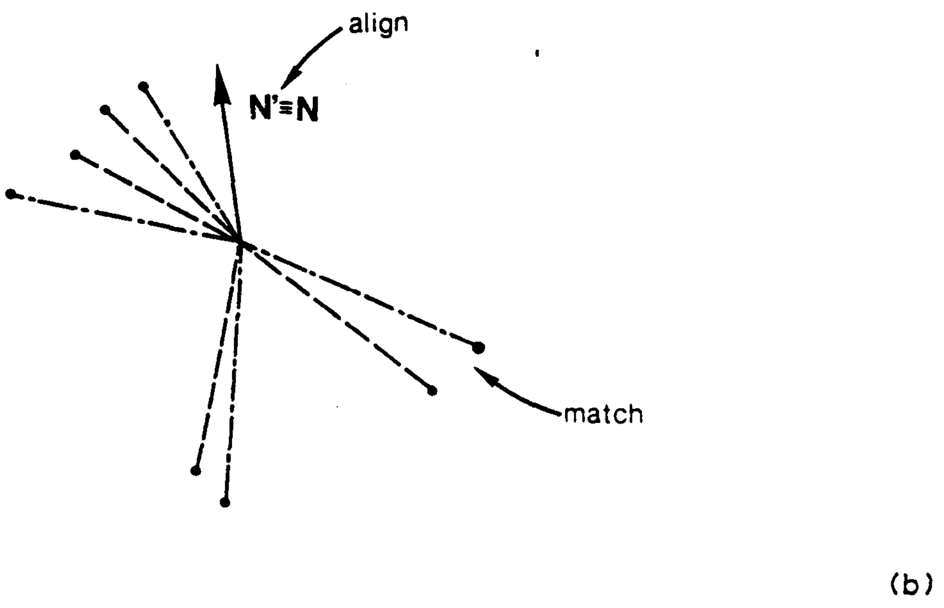


Figure 4.4 Alignment of surface-normal vectors in quad-tree nodes: (a) orientations before alignment, (b) orientations after alignment of surface-normal vectors in top nodes

and $\underline{B}'$, respectively, to planes perpendicular to $\overline{N}$ and $\overline{N}'$, respectively. $\underline{G}$ is then rotated around $\overline{N}$ to match every projected successor of $\underline{B}$ with every projected successor of $\underline{B}'$ and to generate up to four initial surface registrations. Each rotation is found by minimizing the sum of distances between the projected successors of $\underline{B}$ and the projected successors of $\underline{B}'$.

This second type the start operator is implemented by the matching algorithm because it uses only the surface information provided by the quadtree surface representation [Chapter 3]. It does not require detection of special points or features on the two surfaces and their matching or any other additional extraneous processing. If the size of the bounding volumes, aligned by the start operator, is small then a good estimate of the spatial registration, resulting in a short search, is found. This, however, usually causes a large number of initial nodes to be generated.

The expansion operator attempts to improve the spatial registration of the best node present on OPEN in step (5) of the matching algorithm (4.21) by generating its successors with new 3D rigid transformations. It has these two strategies available:

(1) compute $\overline{T}$ from the coordinates of $\overline{h}(x,y,z,w)_i$ and $\overline{h}'(x,y,z,w)_i$ for all $\underline{M}$ valid intersection points; and

(2) modify values of the individual transformation pa-

rameters of $\overline{T}$.

The first strategy generates a successor whose transformation $\overline{T}$ is computed by minimizing the sum of distances of all $\underline{M}$ points $\overline{h}(x,y,z,w)_i$ and their intersections $\overline{h}'(x,y,z,w)_i$ as described in Section 4.2 by equations (4.6-11) and procedure (4.12). Additional successor nodes may also be generated with transformations which are either only translations or rotations. A translation transformation, $\overline{t}$, is computed by minimizing the sum of distances:

$$\min\left[\sum_{i=1}^{N} \left| \overline{h}'_i - \overline{h}_i + \overline{t} \right|^2\right], \tag{4.26}$$

a rotation transformation, $\overline{R}$, is computed by minimizing the sum of distances:

$$\min\left[\sum_{i=1}^{N} \left| \overline{h}'_i - \overline{R}\,\overline{h}_i \right|^2 + \left| \overline{R}^t\,\overline{h}'_i - \overline{h}_i \right|^2\right], \tag{4.27}$$

or by minimizing the sum of angular differences of surface-normal vectors:

$$\min\left[\sum_{i=1}^{N} \left| \overline{R}\,\overline{N}_i \cdot \overline{N}'_i - 1.0 \right|\right]. \tag{4.28}$$

Equations (4.26) and (4.27) are evaluated for the coordinates of points $\overline{h}(x,y,z,w)_i$ and $\overline{h}'(x,y,z,w)_i$; equation (4.28) is evaluated for surface-normal vectors $\overline{N}_i$ and $\overline{N}'_i$ at $\overline{h}(x,y,z,w)_i$ and $\overline{h}'(x,y,z,w)_i$, respectively. Only the $\underline{M}$ valid intersection points are, of course, used to evaluate these equations. The rotation matrices computed from (4.27) and (4.28) are made orthogonal by procedure (4.12). This strategy, therefore, may produce four successor nodes by minimizing spatial differences between the evaluation points of $\underline{G}$ and their intersections on $\underline{G}'$.

The second expansion strategy modifies the values of the individual transformation parameters of $\overline{T}$. The current value of each active parameter is incremented and decremented by a value proportional to the current surface difference $D(\overline{T})$. There are, usually, six active parameters - tx, ty, tz, rx, ry, rz - whose values are modified, thus producing 12 successor nodes n_1 to n_{12} [Figure 4.5]. There are two functions, $\Delta t(e(n))$ and $\Delta r(e(n))$, which compute modifications of the translation and rotation parameters, respectively:

$$\begin{aligned}
&\Delta tx(n_1) = +\Delta t(e(n)) \, |\Delta tx(n)| \\
&\Delta tx(n_2) = -\Delta t(e(n)) \, |\Delta tx(n)| \\
&\cdots\cdot \qquad\qquad\qquad\qquad\qquad\qquad (4.29) \\
&\cdots\cdot \\
&\Delta rz(n_{11}) = +\Delta r(e(n)) \, |\Delta rz(n)| \\
&\Delta rz(n_{12}) = -\Delta r(e(n)) \, |\Delta rz(n)|
\end{aligned}$$

The refinement functions $\Delta t(e(n))$ and $\Delta r(e(n))$ are functions of the value of the evaluation function $e(n)$ of the current node $\underline{n}$. If they halve the parameter increments of the current node then this strategy approximates a binary search. The values of the parameter increments in a start

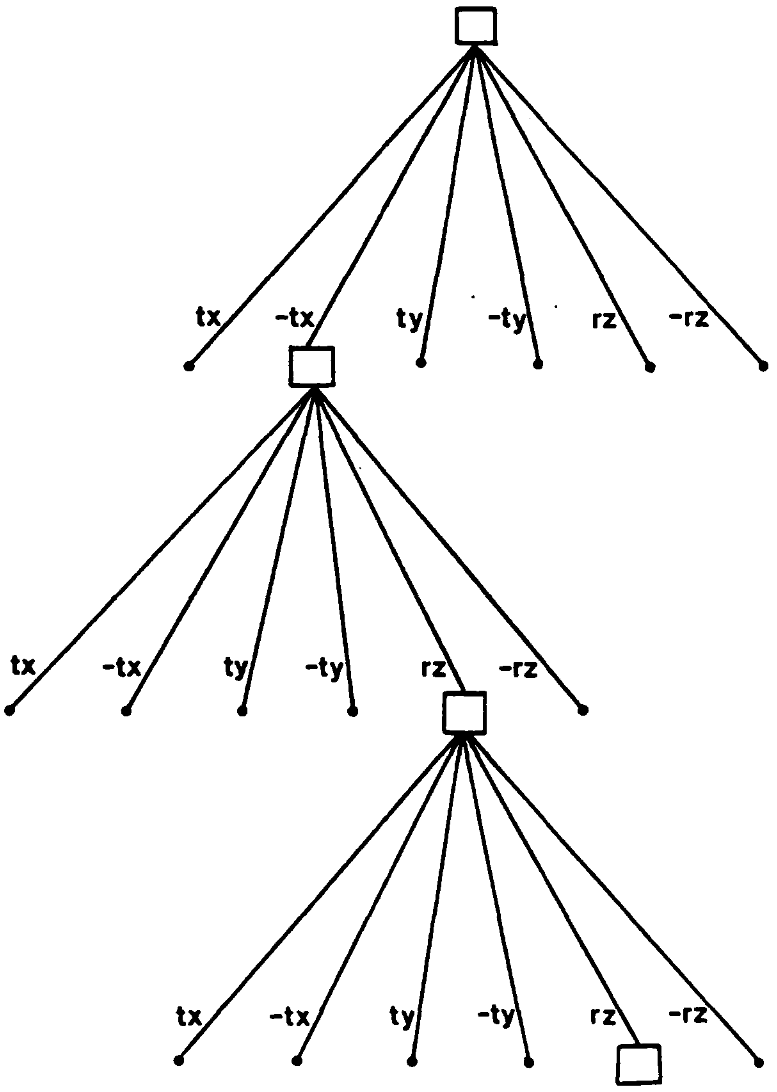


Figure 4.5   Search tree of 3D transformation parameters

node are set to allow the center of a bounding volume $\underline{B}$ to be translated anywhere within $\underline{B}'$, and to allow the surface-normal vector $\overline{N}$ in $\underline{B}$ to be rotated by 90 degrees from $\overline{N}'$ in $\underline{B}'$.

A combination of both strategies is currently employed by the matching algorithm. The second strategy is more effective near the beginning of a search; as the registration improves, the nodes generated by the first strategy converge to the solution substantially faster since each contains modifications of all active parameters rather than only one parameter. Nodes whose transformations are computed only as translations (4.26) or rotations (4.27-28) are usually not as effective as the nodes whose transformations contain both translations and rotations.

The matching algorithm has been able to find the correct spatial registration of complicated multiply-curved surface segments, described by continuous, differentiable surface representations. If the registration of the surfaces is ambiguous or the common (overlapping) surface area is small, the algorithm finds a solution and can be restarted to find further solutions from nodes still present on OPEN.

4.4 Merging Algorithms

The second major problem in assembling complete models of 3D objects from surface segments is that of merging two or more surface descriptions of overlapping segments, defined in the same object coordinate system, into a single surface description, or converting them into a volumetric representation. This section presents several algorithms which:

(a) merge overlapping surface descriptions, given in the same object coordinate system, into a single surface description;

(b) merge overlapping surface descriptions of a closed object, given in the same coordinate system, into a single volumetric representation;

(c) reparametrize a parametric surface representation, or parametrize an algebraic surface representation; and

(d) convert a surface representation of a closed object into a volumetric representation.

During the process of generating a model of an object from surface segments we need to merge two surface descriptions G_k and $G_{k'}$ under these two circumstances:

(a) G_k and $G_{k'}$, originally defined in object coordinate systems $O(x,y,z,w)_k$ and $O(x,y,z,w)_{k'}$ and parameter coordinate systems $P(u,v,w)_j$ and $P(u,v,w)_{j'}$, respectively, have been matched and transformed into $O(x,y,z,w)_{k'}$ and need to be merged into a single parameter coordinate system $P(u,v,w)$ (i.e., during the surface-acquisistion step, the object was moved into a new orientation, which, in effect, also moved the orientation of the parametrizing grid regardless of whether the grid was actually physically moved); or

(b) G_k and $G_{k'}$, defined in the same object coordinate system $O(x,y,z,w)_k \equiv O(x,y,z,w)_{k'}$ but different parameter coordinate systems $P(u,v,w)_j$ and $P(u,v,w)_{j'}$, respectively, need to be merged into the same parameter coordinate system $P(u,v,w)$ (i.e., during the surface-acquisition step, the object remained in the same orientation but the parametrizing grid was moved into a new orientation).

Finally, when all the surface segments of a closed 3D object have been matched and are defined in the same object coordinate system, it is also desirable to merge them into a volumetric representation. The last algorithm in this section converts a surface representation, which consists of overlapping surface segments or a single surface description, into rectangular parallelepiped volumes.

4.4.1 Transformation of Parameters

This method merges two parametric surface descriptions, G_j and $G_{j'}$, parametrized by parameter coordinate systems $P(u,v,w)_j$ and $P(u,v,w)_{j'}$, respectively, by a transformation

[Figure 4.6]:

$$u' = u'(u,v) \tag{4.30}$$

$$v' = v'(u,v)$$

which has the property that the functions u' and v' have continuous partial derivatives, and the transformation can be inverted [22]. A linear transformation from parametric space $P(u,v,w)_j$ to parametric space $P(u,v,w)_{j'}$ is expressed as:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix}_{j'} = T\{P_j \rightarrow P_{j'}\} \begin{bmatrix} u \\ v \\ w \end{bmatrix}_j = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}_j \tag{4.31}$$

The six coefficients of $T\{P_j \rightarrow P_{j'}\}$ can be determined from three non-colinear points $\bar{h}(u,v,w)_j$ in $P(u,v,w)_j$ and their corresponding images $\bar{h}(u,v,w)_{j'}$ in $P(u,v,w)_{j'}$. They are obtained by rewriting (4.31) as two simultaneous linear equations:

$$\begin{bmatrix} u & v & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u & v & 1 \end{bmatrix}_j \begin{bmatrix} t_{11} \\ t_{12} \\ \cdots \\ \cdots \\ t_{22} \\ t_{23} \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix}_{j'} \tag{4.32}$$

or, for short,

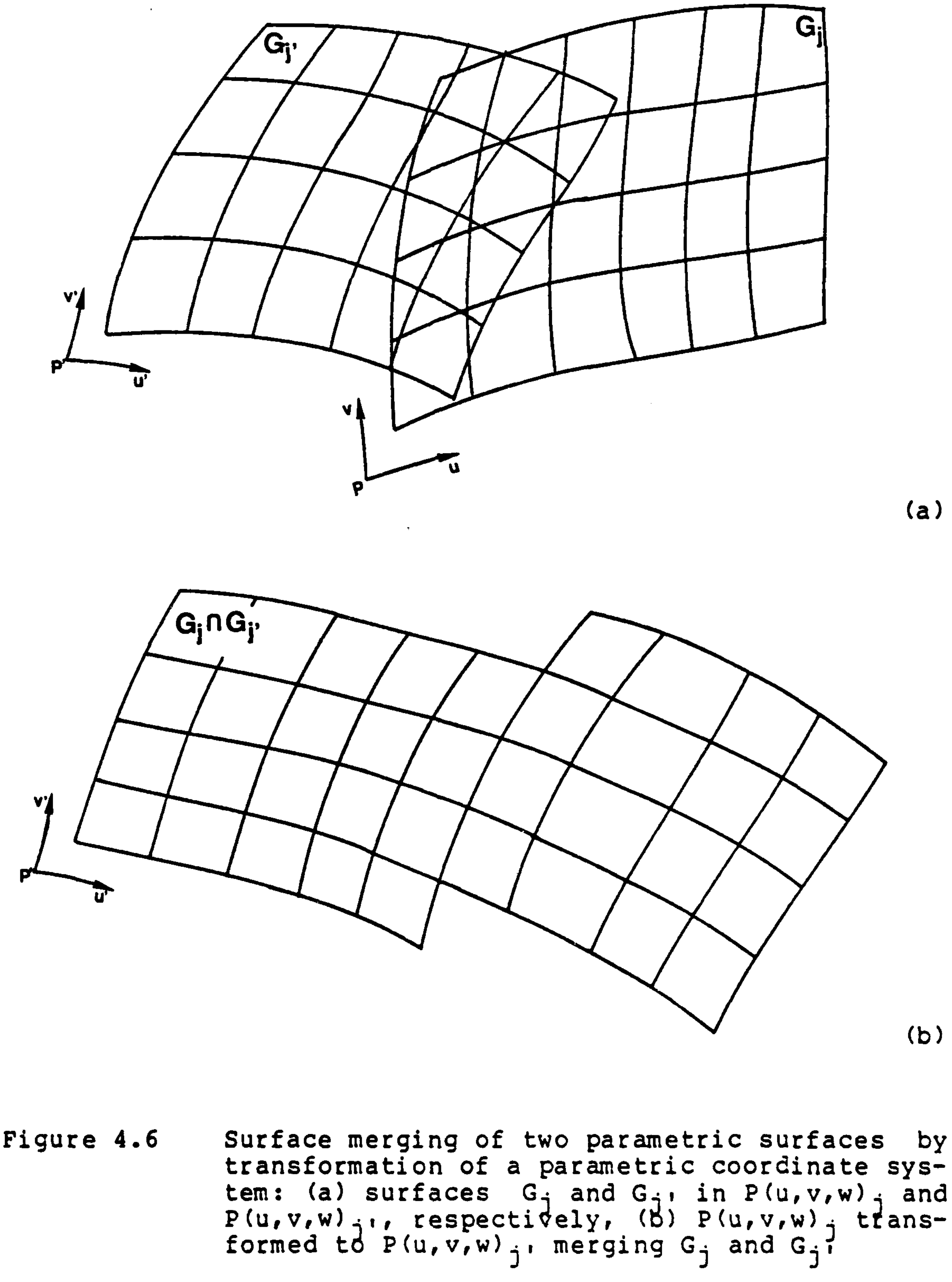


Figure 4.6 Surface merging of two parametric surfaces by transformation of a parametric coordinate system: (a) surfaces G_j and $G_{j'}$ in $P(u,v,w)_j$ and $P(u,v,w)_{j'}$, respectively, (b) $P(u,v,w)_j$ transformed to $P(u,v,w)_{j'}$ merging G_j and $G_{j'}$

$$\bar{D}\,\bar{T} = \bar{D}' \tag{4.33}$$

for each point $\bar{h}(u,v,w)_j$ and its image $\bar{h}(u,v,w)_{j'}$. For $L \geq 3$ points, system (4.33) is again solved by the least-squares method:

$$\begin{bmatrix} \bar{D}_1 \\ \cdot\cdot \\ \bar{D}_L \end{bmatrix}^t \begin{bmatrix} \bar{D}_1 \\ \cdot\cdot \\ \bar{D}_L \end{bmatrix} \bar{T} = \begin{bmatrix} \bar{D}_1 \\ \cdot\cdot \\ \bar{D}_L \end{bmatrix}^t \begin{bmatrix} \bar{D}'_1 \\ \cdots \\ \bar{D}'_L \end{bmatrix} \tag{4.34}$$

The corresponding parametric points in G_j and $G_{j'}$ are again found by the ray-tracing method. At each control point $\bar{h}(m,n)_{j'}$ of $G_{j'}$, where $0 \leq m \leq M$ and $0 \leq n \leq N$, a ray is cast in the direction of the surface normal vector and intersected with G_j. If the nearest intersection point, $\bar{h}(u,v)_j$, is such that

$$\left| \bar{h}(x,y,z,w)_j - \bar{h}(x,y,z,w)_{j'} \right| < \text{epsilon} \tag{4.35}$$

where $\bar{h}(x,y,z,w)_j \equiv \bar{h}(u,v)_j$ and $\bar{h}(x,y,z,w)_{j'} \equiv \bar{h}(m,n)_{j'}$, then the parameter values (m,n) and (u,v) belong to the same surface point, $\bar{h}(x,y,z,w)_j \equiv h(x,y,z,w)_{j'}$, and form equation (4.32). If three or more such pairs of points are found, then the transformation $\bar{T}$ is computed with equation (4.34). The two surfaces [Figure 4.6(a)] are merged by expanding $G_{j'}$ to include G_j. A new set of control points in the $P(u,v,w)_{j'}$ parameter coordinate system is computed on a larger grid $M_{min} \leq m \leq M_{max}$ and $N_{min} \leq n \leq N_{max}$ [Figure 4.6(b)]. A control point $\bar{h}(m,n)$, defined in $G_{j'}$, is directly copied to the new grid. A control point $\bar{h}(m,n)$, not defined in $G_{j'}$, is transformed by $T\{P_{j'} \rightarrow P_j\}$ to $P(u,v,w)_j$ and obtained from G_j. If the point is also not defined in G_j, then it remains undefined in the new grid. This procedure is outlined as follows:

(4.36)

```
procedure merge_transform;
   begin
   [compute transformation]
   L := 0;
   for n := 0 step 1 until M do
      begin
      for m := 0 step 1 until N do
         begin
         make_ray(ray(m,n),G_j');
         intersect(ray(m,n),G_j,point(u,v));
         if point(u,v) valid then
                                 L := L + 1;
                                 match(m,n,u,v,L);
                               end;
         end;
      end;
   if L < 3 then return(fail);
   solve(match,L,T);
   [merge surfaces to a new network]
   for n := $N_{min}$ step 1 until $N_{max}$ do
      begin
      for m := $M_{min}$ step 1 until $M_{max}$ do
         begin
         find(point(m,n),G_j');
         if point(m,n) void then
                                transform(u,v,m,n,T);
                                find(point(u,v),G_j);
                                if point(u,v) valid then
                                  point(m,n) := point(u,v);
                              end;
         output(point(m,n));
         end;
      end;
   return;
   end;
```

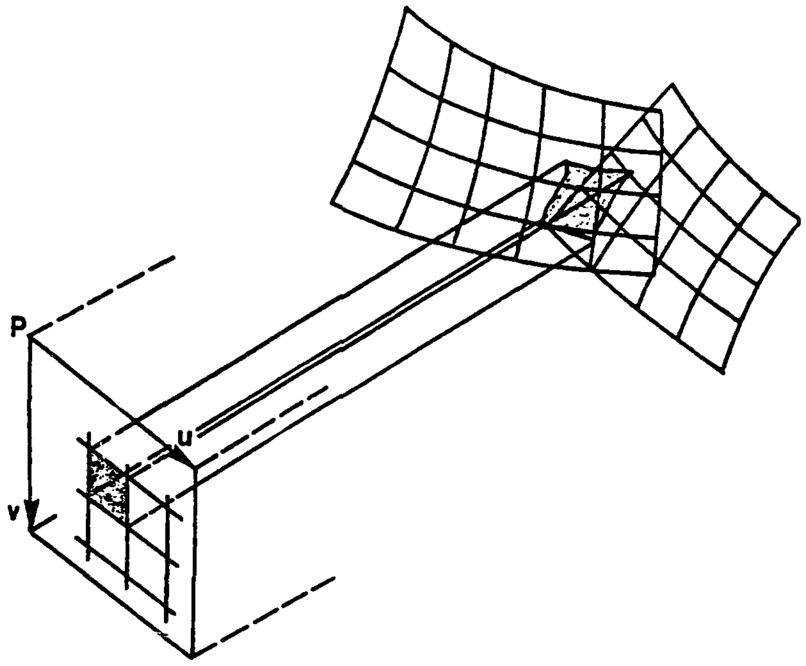
In this method the parameter coordinate system of $G_{j'}$

remains in place while the parameter coordinate system of G_j is transformed, potentially disrupting the parametrization of G_j. The next section contains several algorithms which parametrize both surfaces with a new parameter coordinate system.

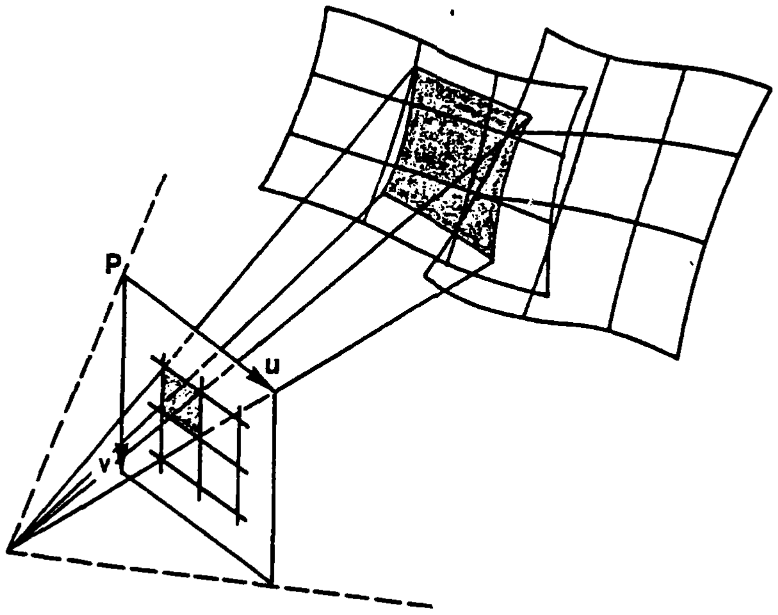
4.4.2 Projection of Parameters

This technique merges two or more parametric surfaces into a single parameter coordinate system by projecting this parameter system on the surfaces [Figure 4.7]. This approach is similar to that used in Chapter 2 to acquire 3D surface data by physically projecting an actual grid. Here the projection is done to surface models in a data base. The projected parameter coordinate system is usually defined by a plane, and the projection is orthographic [Figure 4.7(a)], or perspective [Figure 4.7(b)]. Alternatively, the parameter coordinate system could be defined by a spherical or cylindrical surface surrounding the surfaces to be parametrized. In addition to reparametrizing parametric surfaces, this method can also be used to parametrize a surface defined only by an algebraic representation.

This procedure defines a 2D parameter coordinate system P(u,v,w) in a 3D object coordinate system O(x,y,z,w). At sampling distances Δu and Δv in the plane, rays are cast from the plane to the surfaces; the first intersection of a



(a)



(b)

Figure 4.7 Surface merging by reparametrization: a parameter coordinate system is projected on surfaces with (a) orthographic projection, (b) perspective projection

ray with a surface then becomes a control point on the newly parametrized surface. If overlapping surface segments are being merged, then the nearest intersections of a ray and the different segments, which are within a given tolerance, are averaged into a single intersection point. The intersection point of a ray at $\overline{h}(u,v)$ is connected to the intersection points found by the four adjacent rays at $\overline{h}(u+\Delta u)$, $\overline{h}(u-\Delta u,v)$, $\overline{h}(u,v+\Delta v)$, and $\overline{h}(u,v-\Delta v)$. The algorithm, therefore, defines the usual bivariate network of 3D points, which is subsequently converted into a quadtree of composite bicubic patches:

(4.37)

```
procedure merge_parametrize;
   begin
   make_projection(plane);
   for v := $v_{min}$ step $\Delta v$ until $v_{max}$ do
      begin
      for u := $u_{min}$ step $\Delta u$ until $u_{max}$ do
         begin
         make_ray(ray(u,v),plane);
         intersect(ray(u,v),G,point(u,v));
         connect(point(u,v),point(u-$\Delta$u,v));
         connect(point(u,v),point(u,v-$\Delta$v));
         end;
      end;
   return;
   end;
```

The algorithm is illustrated in Figure 4.8(a-c). A sphere [Figure 4.8(a)] is parametrized by orthographic projection [Figure 4.8(b)], and perspective projection [Figure 4.8(c)] of a planar parameter coordinate system. There are 32 x 32 sample rays cast from the projection plane to the sphere.

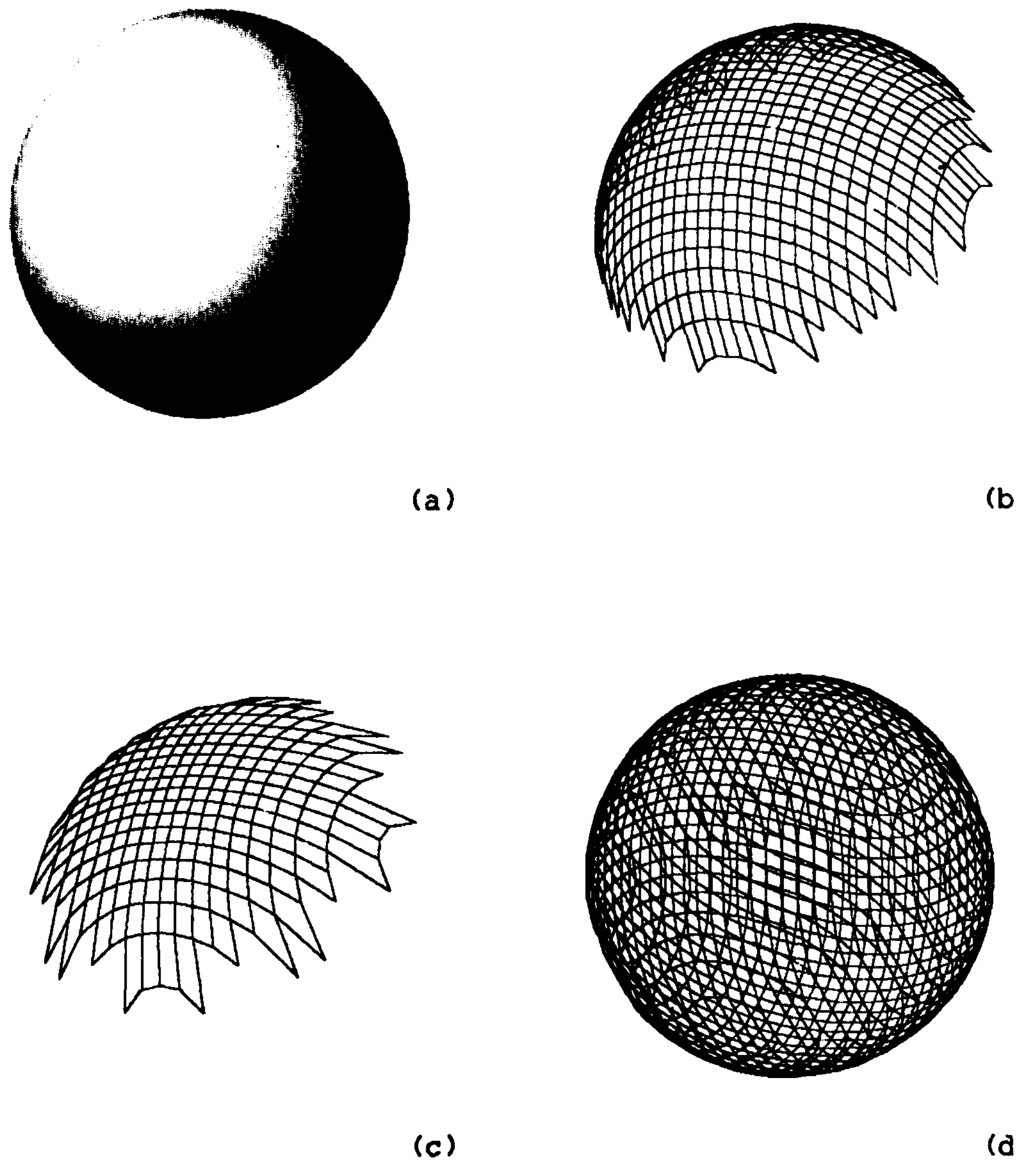


Figure 4.8 Parametrization of a sphere with a projected parameter system: (a) image of a sphere, (b) orthographic projection, (c) perspective projection, (d) closed parametric representation from orthographic projection

Algorithm (4.37) processes only the surface points nearest to the projected plane along a sample ray. For a solid object, as shown in Figure 4.8(b-c), it generates a parametric surface network of only the nearest surface sections facing the projection plane. However, the algorithm can be modified to produce a closed parametric network of the complete object. First, all intersections of a ray and the surfaces of a solid object are computed. There must be 2 $\underline{S}$ such intersections where $\underline{S}$ is the number of line segments of a ray inside the object. Intersection points where a ray is tangential to a surface, and only one intersection point is found, are eliminated by testing the surface-normal vector, $\overline{N}$, at the point of intersection. If $\overline{N}$ is nearly orthogonal, within a tolerance, to the ray then the intersection is deleted. Intersections of overlapping surface segments are again averaged into a single intersection. The second modification of algorithm (4.37) affects the creation of connections of the intersection points located by adjacent rays: if a point on an adjacent ray is closer to the current point than the next point on the current ray then it is connected to the current point; otherwise, the next point on the current ray is connected to the current point. A closed network of 3D points is created by the new algorithm, assuming that the object does not have any extrusions or protrusions thinner than the sampling distances Δu and Δv:

(4.38)

```
procedure merge_close;
   begin
   make_projection(plane);
   for v := $v_{min}$ step $\Delta$v until $v_{max}$ do
      begin
      for u := $u_{min}$ step $\Delta$u until $u_{max}$ do
         begin
         make_ray(ray(u,v),plane);
         intersect(ray(u,v),G,point(u,v,s),N(s),S2);
         delete(ray(u,v),point(u,v,s),N(s),S2,0.010);
         for s := 1 step 2 until S2 do
            begin
            if |point(u,v,s)-point(u,v,s+1)|>
               |point(u,v,s)-point(u-$\Delta$u,v,s)| then
              begin
              connect(point(u,v,s),point(u-$\Delta$u,v,s));
              connect(point(u,v,s+1),point(u-$\Delta$u,v,s+1));
              end;
              else
              connect(point(u,v,s),point(u,v,s+1));
            if |point(u,v,s)-point(u,v,s+1)|>
               |point(u,v,s)-point(u,v-$\Delta$v,s+1)| then
              begin
              connect(point(u,v,s),point(u,v-$\Delta$v,s));
              connect(point(u,v,s+1),point(u,v-$\Delta$v,s+1));
              end;
              else
              connect(point(u,v,s),point(u,v,s+1));
            end;
         end;
      end;
   return;
   end;
```

A closed network of 3D points of a sphere [Figure 4.8(a)], generated by this algorithm, is shown in Figure 4.8(d). There are again 32 x 32 sample rays cast from the param:ter plane.

The previous algorithm (4.38) generated a closed network of 3D surface points of a solid object. While the surface sections approximately parallel to the projection plane were closely sampled (with distances of adjacent points on the order of Δu or Δv), the surface sections approximately orthogonal to the projected plane were coarsely sampled. An improved algorithm samples the object by rays projected from three mutually orthogonal planes [Figure 4.9]. For each projection plane only the intersection points, where the angular difference between a ray and one of the components of the surface-normal vector is less than 45 degrees, are retained. They are connected to the corresponding points found by the adjacent rays, as in algorithm (4.37), and form disjoint networks. After these networks have been obtained for all three planes, they are connected into a single network by joining points in different networks within the sampling distance:

(4.39)

```
procedure merge_orthogonal;
   begin
   for p := 1 step 1 until 3 do
      begin
      make_projection(plane(p));
      for v := $v_{min}$(p) step $\Delta$v until $v_{max}$(p) do
         begin
         for u := $u_{min}$(p) step $\Delta$u until $u_{max}$(p) do
            begin
            make_ray(ray(u,v),plane(p));
            intersect(ray(u,v),G,point(u,v,s),N(s),S2);
            delete(ray(u,v),point(u,v,s),N(s),S2,0.707);
            for s := 1 step 2 until S2 do
               begin
               connect(point(u,v,s),point(u-$\Delta$u,v,s));
               connect(point(u,v,s),point(u,v-$\Delta$v,s));
               connect(point(u,v,s+1),
                       point(u-$\Delta$u,v,s+1));
               connect(point(u,v,s+1),
                       point(u,v-$\Delta$v,s+1));
```

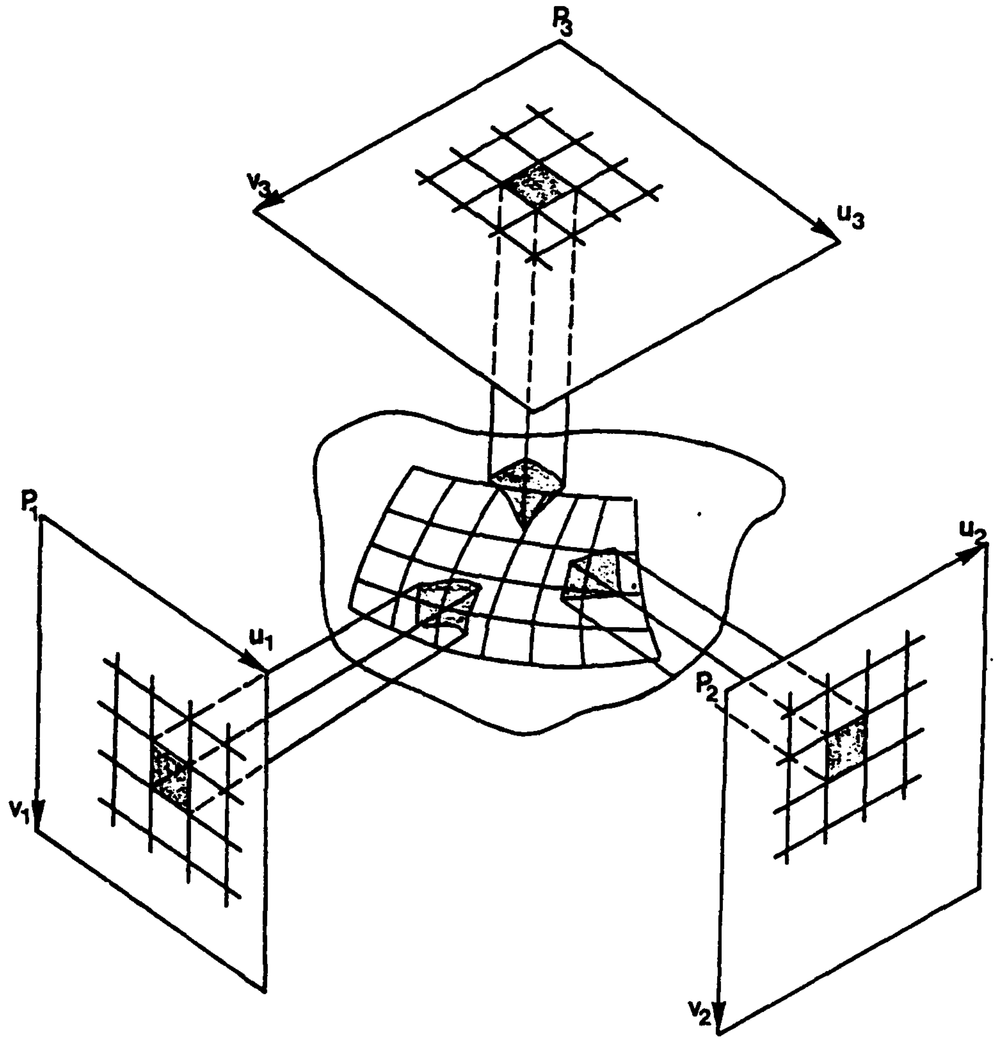


Figure 4.9 Surface merging by reparametrization: three mutually orthogonal parameter coordinate systems are orthographically projected on the surfaces of a closed object

```
                end;
              end;
            end;
          end;
        create_networks(point);
        return;
        end;
```

This approach is illustrated with a sphere [Figure 4.8(a)] in Figure 4.10. The three orthogonal planes are projected to generate the top and bottom [Figure 4.10(a)], front and back [Figure 4.10(b)], and left and right [Figure 4.10(c)] networks which are then connected into a single network [Figure 4.10(d)]. There were 24 x 24 sample rays cast from each projection plane.

The last two algorithms (4.38 and 4.39) can produce degenerate surface regions bounded by three or six parametric curves. Such regions need to be specially processed when the parametrized surface is being converted into topologically rectangular patches. A region bounded by three curves is handled as a degenerate, topologically triangular patch. A region bounded by six curves is split into two topologically rectangular patches.

In order that algorithms (4.37-39) generate 3D surface data consistent with the data provided by the photogrammetric reconstruction method [Section 2.4], the surface-normal vector is computed at each control point of a parametric network. Normal vectors of the surface of a solid object are oriented to the outside of the object, normal

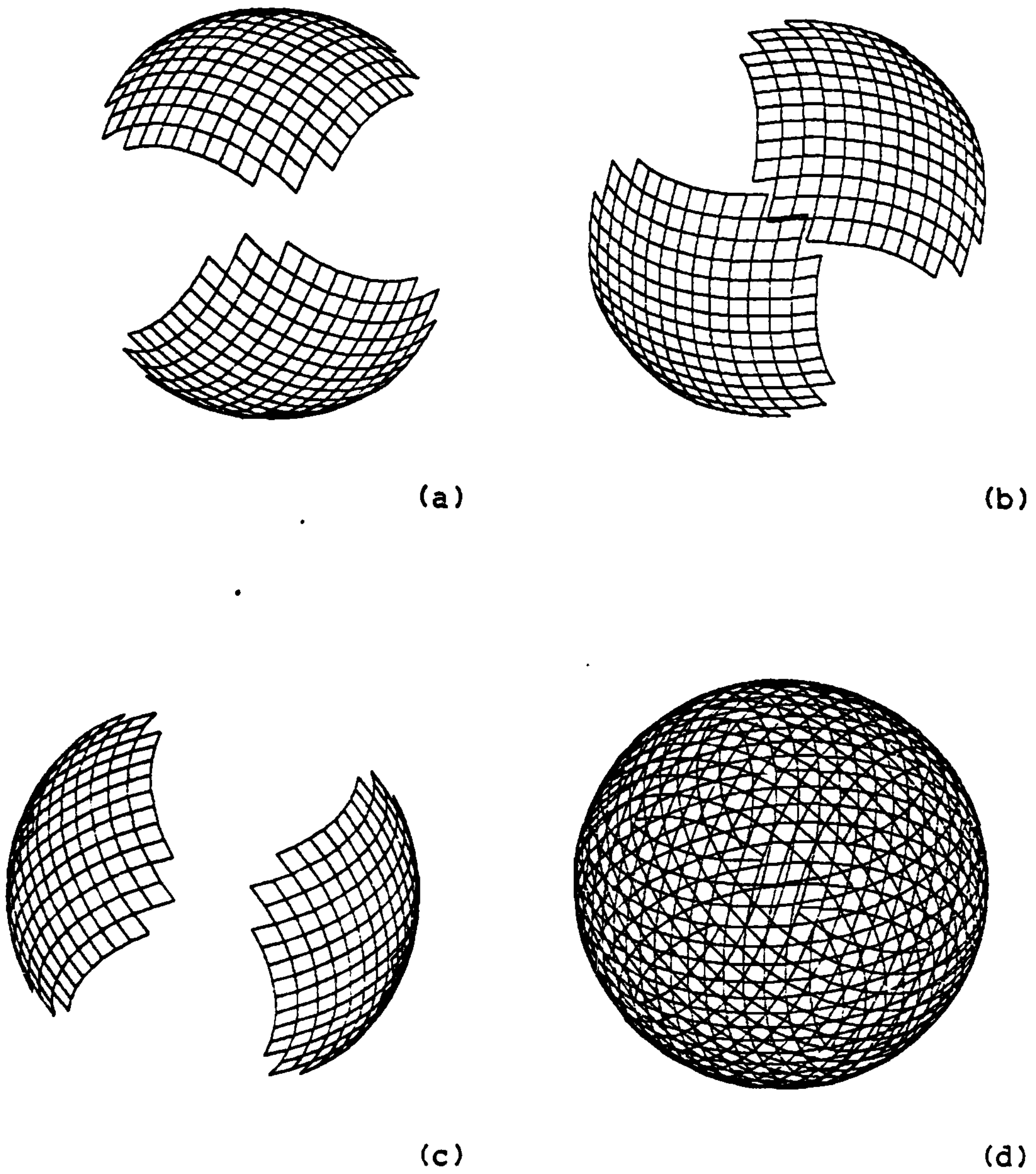


Figure 4.10 Parametrization of sphere by three orthogonal parameter coordinate systems: (a) top and bottom, (b) front and back, (c) left and right networks, (d) all networks connected

vectors of other surfaces are oriented towards the projection plane.

4.4.3 Conversion to Polyhedra

The last merging algorithm converts either a set of overlapping surface segments or a single surface description, both of which enclose a solid object, into a volume representation made of parallel parallelepipeds. The algorithm partitions the object into parallelepipeds by intersecting parallel rays cast from a projection plane with the surfaces of the object. All intersections of a ray with the object are computed as in algorithm (4.38). Each line segment of a ray inside the object defines a parallelepiped centered around the segment with a Δu x Δv cross-section [Figure 4.11]:

(4.40)

```
procedure merge_solid;
   begin
   make_projection(plane);
   for v := $v_{min} + \Delta v/2$ step $\Delta v$ until $v_{max}$ do
      begin
      for u := $u_{min} + \Delta u/2$ step $\Delta u$ until $u_{max}$ do
         begin
         make_ray(ray(u,v),plane);
         intersect(ray(u,v),G,point(u,v,s),N(s),S2);
         for s := 1 step 2 until S2 do
            begin
            parallelepiped(point(u,v,s),point(u,v,s+1),
                           u,$\Delta$u,v,$\Delta$v,N(s),N(s+1));
            end;
         end;
      end;
```

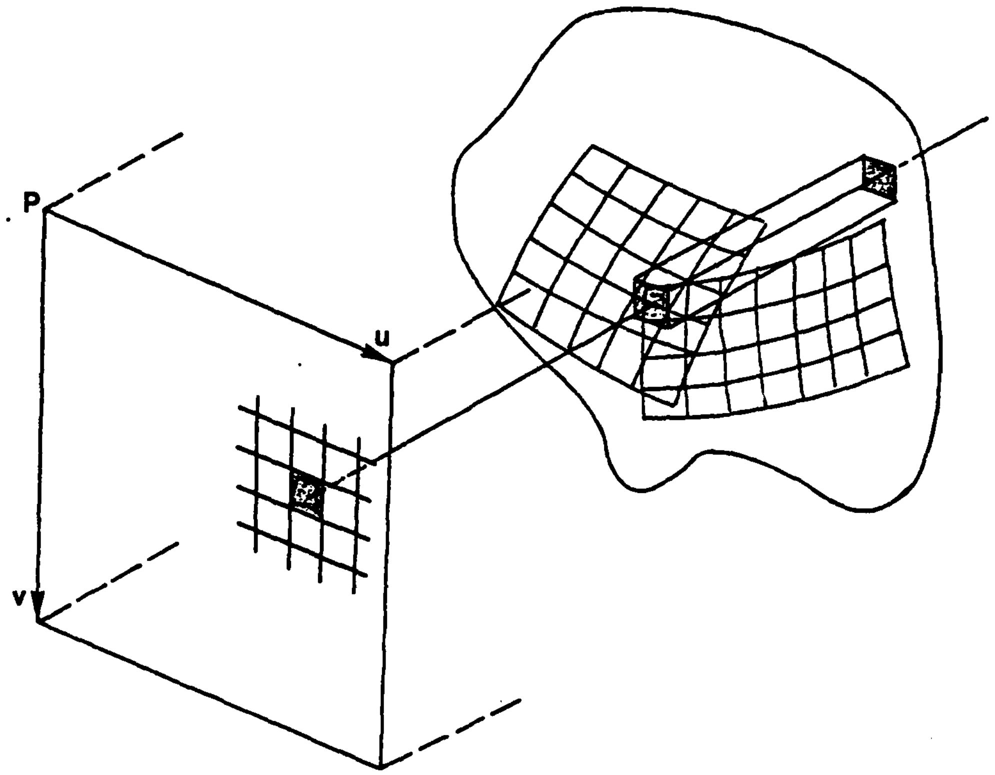


Figure 4.11 Merging of surfaces of a solid object and conversion into rectangular parallelepipeds

```
return;
end;
```

The algorithm evaluates the surface-normal vector at each intersection of a ray with the object and attaches it to the parallelepiped. The normal vector is oriented to the outside of the solid. This representation of a solid object is further converted into the octree representation [44] to allow efficient analysis and shaded display.

This algorithm is illustrated with a sphere in Figure 4.12. The line segments of 16 x 16 sample rays inside the sphere are shown in Figure 4.12(a). The 16 x 16 completed parallelepipeds are shown in Figure 4.12(b). These parallelepipeds, converted to an octree at 16 x 16 x 16 cube resolution, are shown as a shaded image in Figure 4.12(c). The shading is computed with surface-normal vectors generated by the conversion algorithm (4.40) from the algebraical representation of the sphere and attached to each cube. A higher-resolution octree representation of the sphere, at 64 x 64 x 64 cubes, is shown in Figure 4.12(d).

The advantages of using an analytical model with a parametric or algebraic surface representation and converting it to a polyhedra representation are: (a) the analytical model is more compact, (b) arbitrary precision of the conversion is available, and (c) analysis and display of the polyhedra model are usually faster.

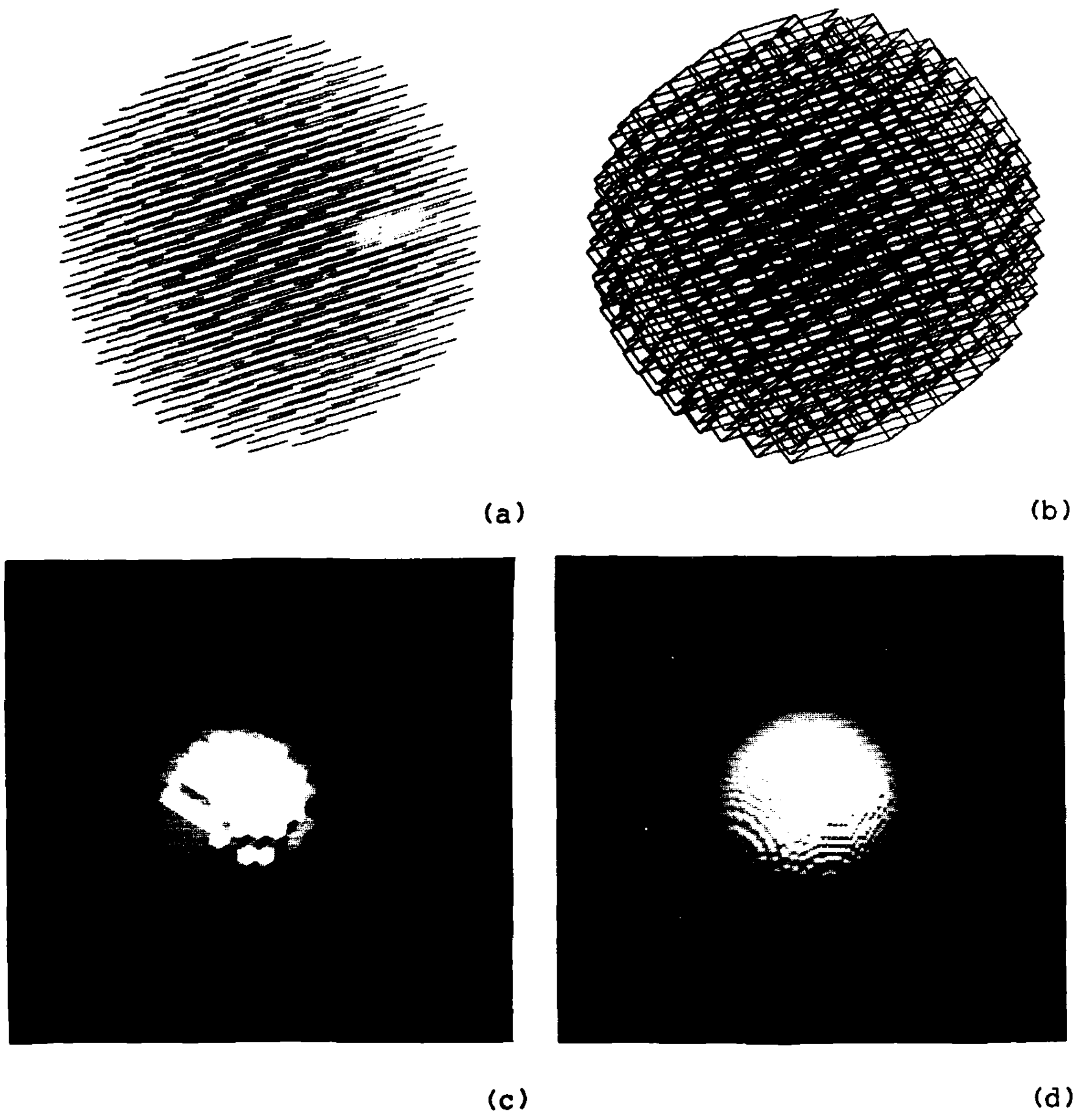


Figure 4.12 Representation of a sphere by parallelepipeds: (a) inside ray segments (16 x 16 rays), (b) completed parallelepipeds (16 x 16 parallelepipeds), (c) low-resolution octree (16 x 16 x 16 cubes), (d) higher-resolution octree (64 x 64 x 64 cubes)

4.5 Applications of the Matching Algorithm

This section describes applications of the matching algorithm to these tasks: (1) surface matching to build complete models of objects from surface segments, (2) surface recognition by matching a surface description with a set of objects in a library of objects, and (3) surface matching to segment an object into surface or volume primitives.

4.5.1 Object Modeling

Using the surface acquisition method developed in Chapter 2 and the surface matching and merging techniques developed in this chapter we assemble complete models of the measured objects. Each modeled object is measured in a number of stable positions; in each stable position it is illuminated in several parameter positions; in each stable and parameter position it is photographed from two or more camera positions. The set of $\underline{K}$ stable positions in which an object is photographed is denoted by:

$$O(x,y,z,w)_1,\ \ldots,\ O(x,y,z,w)_k,\ \ldots,\ C(x,y,z,w)_K.$$

In a stable position $\underline{k}$ the object is photographed in $\underline{J}_k$ parameter positions denoted by:

$$P(u,v,w)_{1,k},\ \ldots,\ P(u,v,w)_{j,k},\ \ldots,\ P(u,v,w)_{J_k,k}.$$

Furthermore, in stable position $\underline{k}$ and parameter position $\underline{j}$ the object is photographed from $\underline{I}_{j,k}$ camera positions, denoted by:

$$Q(r,s,w)_{1,j,k},\ \ldots,\ Q(r,s,w)_{i,j,k},\ \ldots,\ Q(r,s,w)_{I_{j,k},j,k}.$$

The total number of parameter positions, $\underline{J}$, and the total number of images, $\underline{I}$, are given by:

$$J = \sum_{k=1}^{K} J_k, \quad \text{and} \quad I = \sum_{k=1}^{K} \sum_{j=1}^{J} I_{j,k}. \tag{4.41}$$

In each parameter position $\underline{j}$ there is a 3D surface segment reconstructed from $\underline{I}_{j,k}$ images. All surface segments, given in the same stable position $\underline{k}$, are directly merged into a single surface segment. These merged surface segments are then sequentially matched and merged into the complete model. The matching procedure (4.21) is set to the "partially-overlapping-surfaces" mode [Section 4.2] to match surface segments of the same object. Normally, the surface segments being matched are assumed to have been obtained from different stable positions of the object, and the algorithm uses six active transformation parameters - $\underline{tx}$, $\underline{ty}$, $\underline{tz}$, $\underline{rx}$, $\underline{ry}$, $\underline{rz}$ - of the general transformation (4.1). However, if there is à priori knowledge that the segments were obtained from the same stable position of the object then the algorithm uses only three active parameters - $\underline{tx}$, $\underline{ty}$, $\underline{rz}$ - of the limited transformation (4.2). When all surface segments of a solid object have been matched they are merged into a single surface description with algorithm (4.39) and a volume description with algorithm (4.40). The entire modeling process is outlined in the following algorithm:

(4.42)

```
procedure model;
   begin
   for k := 1 step 1 until K do
      begin
      for j := 1 step 1 until $J_k$ do
         begin
         for i := 1 step 1 until $I_{j,k}$ do
            begin
            obtain surface information from image $Q_{i,j,k}$;
            end;
         reconstruct surface segment j;
         if j > 1 then
            merge surface segments j and j-1 into j;
         end;
      if k > 1 then
         begin
         match surface segments k-1 and j;
         merge surface segments k-1 and j into k;
         end;
      else
         begin
         assign surface segment j to k;
         end;
      end;
   merge_orthogonal;
   merge_solid;
   return;
   end;
```

Experimental results of this modeling process are illustrated in Chapter 6.

4.5.2 Surface Recognition

The next proposed application of the matching algorithm is to surface recognition. Here, the algorithm matches a 3D surface segment of an object obtained from a single vantage point with a set of complete models stored in a data base.

The purpose of such matching is (1) to identify the object, and (2) to determine its orientation. The data base, where models are stored, is organized as a single surface graph [Chapter 3], which is partitioned into groups of similar objects using the relational-connection nodes [Section 3.2]. Once the recognizing procedure determines that a partition of objects may contain the surface segment, it visits all the objects within the partition and attempts to match them with the segment. The matching algorithm operates in the "completely-overlapping-surfaces" mode [Section 4.2] and uses the general 3D transformation (4.1):

(4.43)

```
procedure recognize(SEGMENT,LIBRARY);
   begin
   visit the next partition R-node in LIBRARY;
   if not found then return(fail);
   if PARTITION similar to SEGMENT then
           begin
           visit each OBJECT in PARTITION;
           if match(OBJECT,SEGMENT,$\overline{T}$) then return($\overline{T}$);
           end;
   recognize(SEGMENT,R-node);
   return(fail);
   end;
```

This approach generates the orientation of the complete object from the surface segment, and is, therefore, pertinent to the 3D manipulation of the actual object.

4.5.3 Surface and Volume Segmentation

The last proposed application of the matching algorithm is to object segmentation. It is often necessary to segment a 3D numerical surface model into sections that have common shape features or other geometric properties before high-level tasks can use such a model. It is also desirable to find structure relationships among these segmented sections. There are two approaches to the segmentation process described here.

In the first approach, the 3D models are segmented into surface primitives. A surface primitive is designed from one or more surface elements of any available surface representation. It should contain meaningful shape properties of the model that is to be segmented. The primitives are designed at a standard orientation and scale. The evaluation points, where the shape difference (4.16) between the primitive and a model is to be computed, can be selected at this stage. The matching algorithm then matches the designed primitives with the model. The algorithm uses the "completely-overlapping-surfaces" mode and the general 3D transformation. Since the primitives are specified at a standard scale, the matching algorithm must also use scale factors as active transformation parameters, i.e., the transformation is not truly rigid anymore. The primitives are processed according to their priorities; when a match of a primitive and a model is found the surface of the primitive is deleted from the model and the process is repeated with the same primitive until a match cannot be found, then the next primitive is processed:

(4.44)

```
procedure segment_surface(SURFACE_PRIMITIVE,$P_{max}$,OBJECT);
   begin
   for p := 1 step 1 until $P_{max}$ do
      begin
      while match(OBJECT,SURFACE_PRIMITIVE(p),$\overline{T}$) do
         begin
         delete(OBJECT,SURFACE_PRIMITIVE(p),$\overline{T}$);
         output(SURFACE_PRIMITIVE(p),$\overline{T}$);
         end;
      end;
   return;
   end;
```

A union of the segmented primitives constitutes the surface model of the object.

In the second approach, surface models of solid objects are segmented into volumetric primitives. A volumetric primitive is designed as a union of one or more surface elements which completely enclose a 3D space. The matching algorithm spatially registers the surfaces of the primitive with the surfaces of the object. At each evaluation point of a primitive, the half-space (inside or outside), where surface difference (4.16) is to be computed, is specified. Therefore, a volumetric primitive can be spatially registered to (a) enclose completely, (b) be enclosed completely, (c) be completely outside, or (d) be partially outside the solid object. The union, intersection, and difference set operators can be used to construct a CSG tree [Section 3.2] from the segmented primitives. The matching algorithm uses the "partially-overlapping-surfaces" mode and the general 3D transformation with scale factors. The volume-segmentation process is similar to algorithm (4.44):

(4.45)

```
procedure segment_volume(VOLUME_PRIMITIVE,$P_{max}$,
                         SOLID_OBJECT);
   begin
   CSG_TREE := empty;
   for p := 1 step 1 until $P_{max}$ do
      begin
      while match(SOLID_OBJECT,VOLUME_PRIMITIVE(p),$\overline{T}$) do
         begin
         delete(SOLID_OBJECT,VOLUME_PRIMITIVE(p),$\overline{T}$);
         interference(VOLUME_PRIMITIVE(p),$\overline{T}$,SOLID_OBJECT,
                      operator);
         attach(VOLUME_PRIMITIVE(p),$\overline{T}$,operator,CSG_TREE);
         end;
      end;
   return(CSG_TREE);
   end;
```

The union-set operator is applied to primitives completely inside the solid; the intersection operator is applied to primitives partially inside the solid; and the difference operator is applied to primitives completely outside the solid.

4.6 Illustrations

The first example illustrates the matching algorithm by matching two surface segments [Figure 4.13]. Surface segment G_1 is defined in object coordinate system $O(x,y,z,w)_1$ [Figure 4.13(a)], and surface segment G_2 is defined in object coordinate system $O(x,y,z,w)_2$ [Figure 4.13(b)]. Surface segment G_1 has been spatially registered with surface

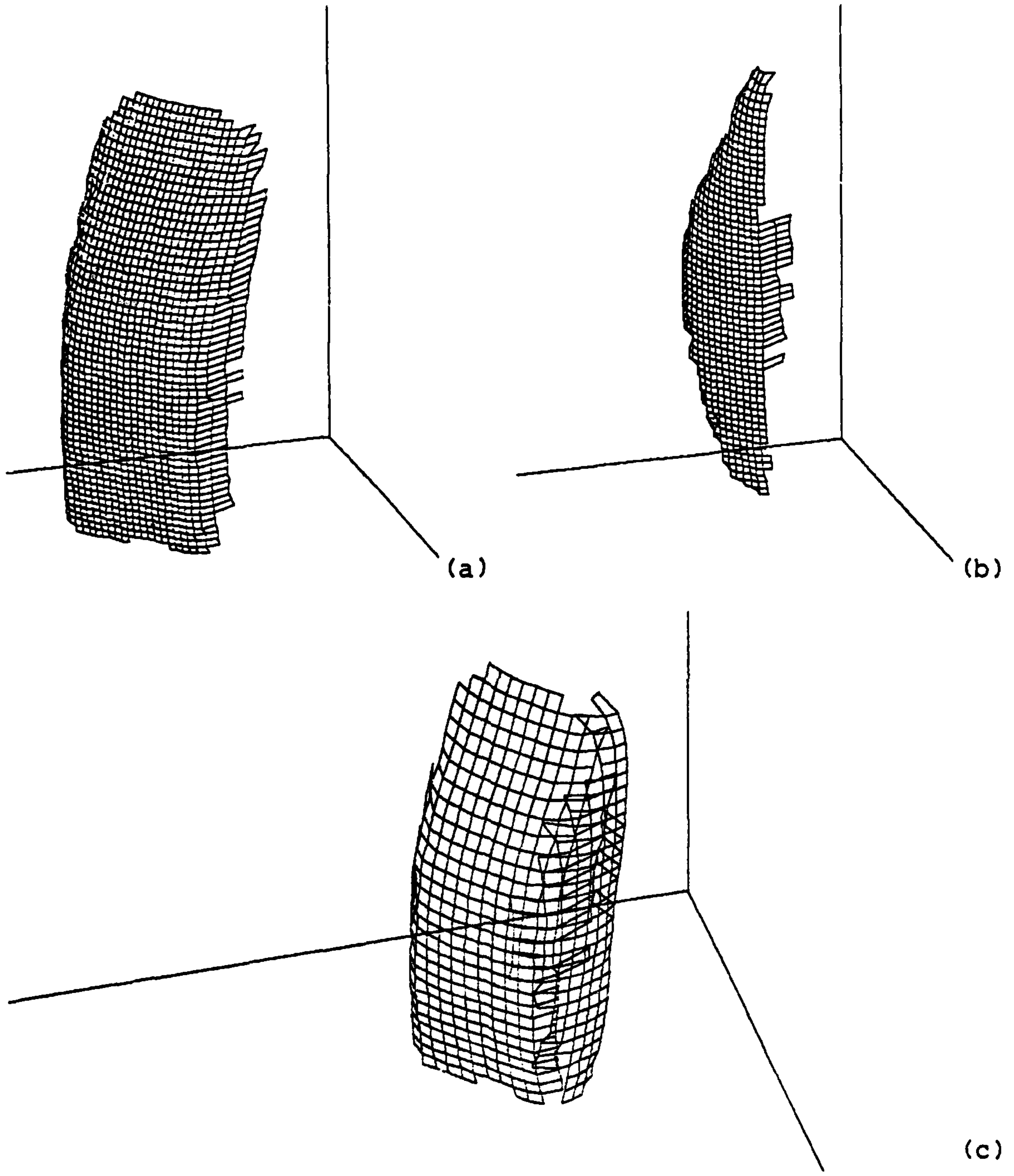


Figure 4.13 Surface matching: (a) surface segment G_k in $O(x,y,z,w)_k$, (b) surface segment $G_{k'}$ in $O(x,y,z,w)_{k'}$, (c) matched surface segments G_k and $G_{k'}$ in $O(x,y,z,w)_{k'}$

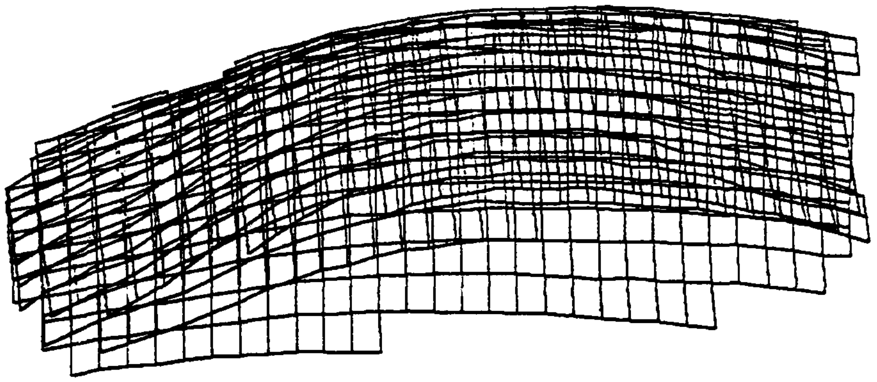
segment G_2 and transformed to $O(x,y,z,w)_2$ [Figure 4.13(c)]. The rigid 3D transformation which aligned the two segments is:

$$T\{O_1 \rightarrow O_2\} = \begin{bmatrix} 0.235 & 0.[illegible]99 & -0.323 & 15.817 \\ 0.723 & 0.213 & 0.658 & 15.043 \\ 0.650 & -0.532 & -0.542 & 69.560 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix} \qquad (4.46)$$

The matching algorithm generated 168 start nodes by aligning surface normal vectors in 9 bounding volumes of G_1 with surface normal vectors in 6 bounding volumes of G_2. These start nodes were expanded into 449 nodes before the solution was found.

The next example illustrates merging of surface segments using the second merging algorithm which projects a new parameter coordinate system [Figure 4.14]. There are three surface segments, G_1, G_2, and G_3, defined in the same object coordinate system $O(x,y,z,w)_1$, and three different parameter coordinate systems $P(u,v,w)_{1,1}$, $P(u,v,w)_{2,1}$, and $P(u,v,w)_{3,1}$ [Figure 4.14(a)]. A square ($\Delta u = \Delta v$) parameter coordinate system was projected on these segments with orthographic projection resulting in a single surface segment [Figure 4.14(b)].

The last example illustrates the merging algorithm (4.40) which converts a surface representation of a solid object into a volume representation [Figure 4.15]. The sur-


(a)

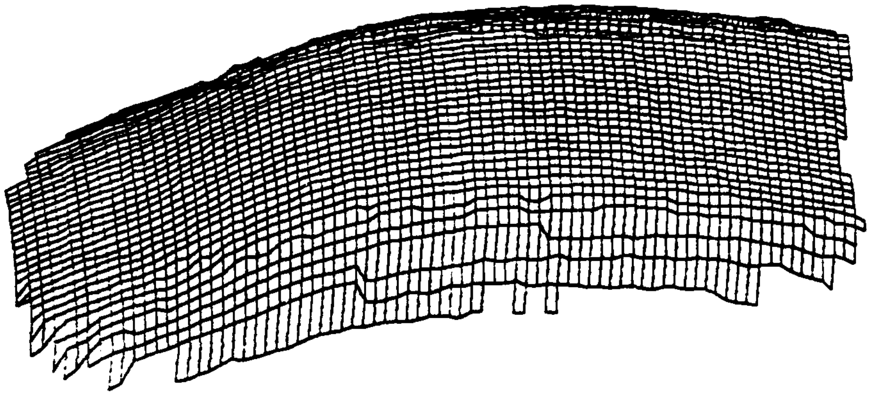
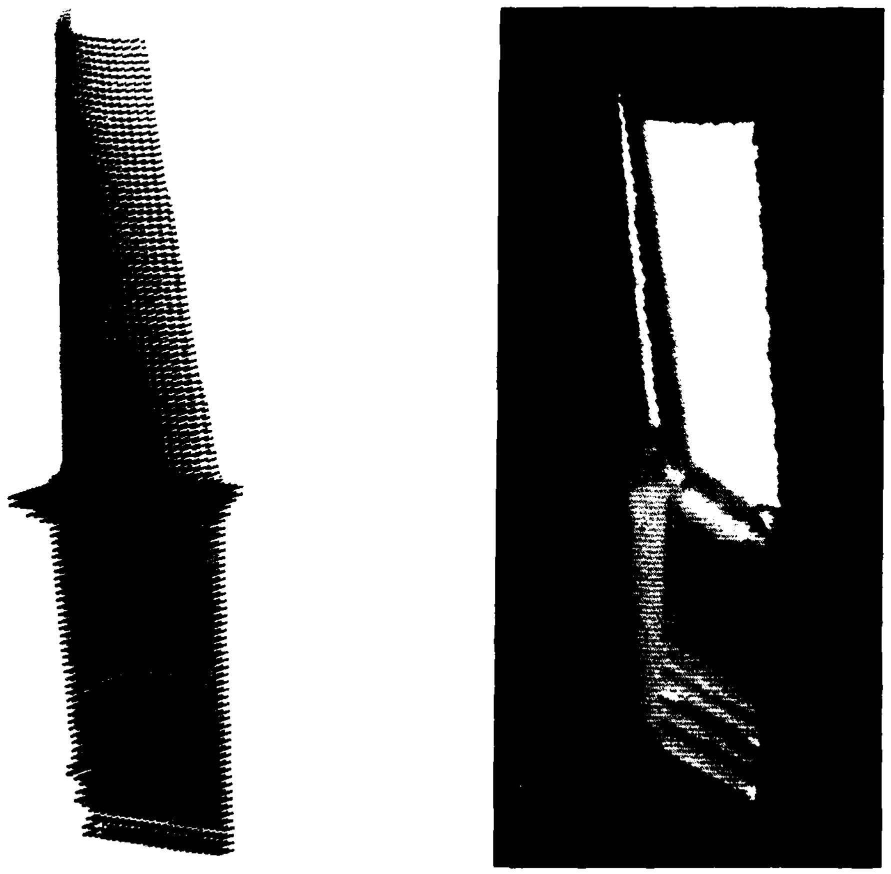

(b)

Figure 4.14 Surface merging: (a) three surface segments parametrized by three parameter coordinate systems, (b) reparametrized into a single parameter coordinate system



(a) (b)

Figure 4.15 Conversion of representation of a J79 turbine blade to (a) parallelepipeds, (b) octree

face model of the J79 turbine blade [Figures 3.10, 3.11 and 3.15] was partitioned into parallelepiped volumes by intersecting the model with orthogonally projected rays. A sampling resolution of 40 x 160 rays converted the blade into 2706 parallelepipeds. The line segments of each ray inside the blade are shown in Figure 4.15(a). The parallelepipeds were further converted into the octree representation. The octree model shown in Figure 4.15(b) contains 35808 nodes, of which 6132 nodes are full. The shading was computed using the surface-normal vectors obtained from the analytical model.

4.7 Summary

A surface matching algorithm which spatially registers 3D surfaces was described in this chapter. The algorithm, in conjuction with several merging algorithms, is used to assemble complete models of 3D objects from their multiple surface segments. This process is illustrated in Chapter 6. Suggested modification of this algorithm for object recognition, or surface and volume segmentation were also presented.

CHAPTER 5

EXPERIMENTAL METHODS

This chapter describes the hardware used and the software developed, in the course of this work, to generate the illustrative examples shown in the three previous chapters and the complete models that will appear in the next chapter.

5.1 Hardware

The photogrammetric process, described in this report, required a camera calibration stand, a projector, a camera and a film scanner. A camera calibration stand was built from three sheets of glass placed perpendicularly to each other. The glass sheets defined the object coordinate system O(x,y,z,w), in which all surfaces were measured. There were ten 30 x 30 mm calibration marks placed in the stand, five each in the x = 0 and y = 0 planes, all clearly visible to the camera. The corners of these marks were detected in the generated images and used to compute the camera transformation matrices $T\{O \rightarrow Q\}$. The size of the stand and the placement of the marks allowed measurement of objects up to 300 x 300 x 300 mm in size. A 35 mm slide projector illuminated the camera calibration stand with a pattern of isoparametric lines defined in a parameter coordinate system P(u,v,w). The pattern was recorded by a photo-densitometer on a 35 mm slide. There are five 20-micron wide lines per mm in both _u_ and _v_ directions of the pattern. These lines cover only a 20 x 20 mm area of the 24 x 36 mm slide to allow normal illumination of the camera calibration marks. A 35 mm camera with a 55 mm focal-length lens was used to photograph the measured surfaces on a high-contrast, black-and-white film with high resolving power. These images were digitized, in Q(r,s,w) image coordinate systems, by the photo-densitometer which sampled the film at 20-micron intervals (50 samples per mm) in both _r_ and _s_ directions. A 16-bit film density value was obtained at each sampled point. In these images, all the relevant information (illuminated surfaces and calibration marks) is contained within a 24 x 30 mm area of the 24 x 36 mm image frame. This means that each image was digitized to a 1200 x 1500 16-bit pixel resolution.

All the data processing was done by a 32-bit midi-computer with (what appears to be at this time) a virtually unlimited virtual memory and a large disk space. A vector drawing and a color raster terminals were used for display of graphical and image data generated by this work. The vector drawing terminal can display up to 8000 2D vectors or 5500 3D vectors from its refresh memory. Orthographic projections of the 3D vectors, following scale, rotation and translation transformations, are computed by the terminal in real time, using special hardware. A 3D surface description can thus be displayed while smoothly "tumbling" on the screen, thereby increasing the viewer's perception of its 3D shape. A hard copy of the image displayed on the screen can be made by an electrostatic printer-plotter with resolution of about 8 binary dots per mm (see, for example, Figures 3.8, 3.10, and 3.12). The raster terminal displays a digital color image from a 512 x 512 x 24 bit frame buffer. It was used here to display windows in the digitized black-and-white images of the measured surfaces as well as synthetic images computed from the 3D surface models. Although the synthetic images cannot be computed in real time, the viewer's perception of the 3D scene shown in a single frame can be augmented by the use of color shading, shadows and textures. A black-and-white and a color hard-copy units are attached to this terminal (see, for example, Figures 3.9, 3.11, 3.13).

5.2 Software

In addition to the development of the surface modeling process, an important secondary goal of this work was that the design and implementation of the process be achieved with good-quality software. There were two key objectives in designing the software:

(1) flexibility - to allow easy modification and expansion, since the system is primarily experimental; and

(2) usability - to allow large quantities of various types of data to be processed without increased effort.

The software was implemented as four command processors. The first three processors contain the modeling procedures described in Chapters 2, 3, and 4, respectively. The fourth processor computes synthetic images from 3D surface data generated by the modeling process and from additional parameters which enhance the 3D scene (light illumination, shadows, textures), and define the viewing camera and its optical system.

A command processor is an interactive software program which allows a user to enter, process, and store data with a set of commands and their parameters. It creates its own software environment, which is required for the given application and is separate from the operating system. Unlike an ordinary program which has its pre-defined input, processing, and output parts, a command processor can execute a flexible sequence of commands appropriate for the data being processed. In the context of our experimental systems, it permits us to display the results of each computational step, and if necessary to repeat the step with different parameters.

The implemented processors' names and tasks are as follows:

<u>SURE</u> - reconstructs 3D surface descriptions from multiple images of the surface illuminated by iso-parametric lines;

<u>SUED</u> - converts 3D surface descriptions into hierarchical representations, enters and edits surfaces, and attaches surface properties;

<u>ALIGN</u> - aligns two surface representations into a common 3D object coordinate space, and optionally merges parametric surface representations into a common 2D parameter coordinate space;

<u>STRAW</u> - generates shaded synthetic images of surface descriptions.

The primary motivations in the software design are listed below:

(a) <u>Common data structures</u>

There is a single definition of all the data structures and parameters within each processor; this definition applies to all the routines in the processor. When a new routine is added to a processor the data base definitions are simply inserted into the new module. Because the processors pass data in the same format to each other, there are large parts of the data structures also shared among different processors.

(b) Uniform storage and access of data files

A data-base management system [55] was developed to maintain all the types of data used in this work, from monochrome and color images to structured models of 3D objects, in hierarchically structured files. Redundant input-output processing and routines are eliminated by maintenance of all the data in a processor's data structures, and by the use of a single input and output routine in each processor, respectively.

(c) Command environment with common syntax and an abbreviation facility

There is common table-driven, command-handling facility in all the processors which prompts for commands, parses them, executes them, and handles syntax errors. Each processor contains a separate control table which contains its prompting strings, commands, keywords, and parameter lists.

Another table in each processor contains symbolic names of data in the processor's data structures. An entry in this table consists of a symbolic name, the data type, and a pointer to the data. Each data item, as it is loaded from a file, generated by a processor, or entered by a command, is given a symbolic name that is used to refer to it in subsequent commands.

An abbreviation facility [55], common to all the processors, allows every non-numeric string to be abbreviated to the leading non-ambiguous substring. The abbreviation facility uniformly applies to all command names, keywords, directory and file names, and symbolic data identifiers.

(d) <u>Modularized software development</u>

The developed software modules are grouped into these categories:

(1) control modules - control execution of a processor, contain command processing;

(2) execution modules - execute data processing commands;

(3) utility modules - provide various utility functions of a processor;

(4) image processing modules - form a library of routines for access and operations on 2D image data;

(5) geometrical processing modules - form a library of routines for access and operations on 2D or 3D geometrical data;

(6) input-output modules - interface to the database management subsystem.

The following sections describe the individual command processors and any associated software in more detail; Appendix D contains the syntax of the actual commands.

5.2.1 Processor SURE

SURE is the surface-reconstruction processor. It implements the techniques described in Chapter 2. The input to this processor is a set of digitized images of the measured surfaces. The surfaces in all such images are in the same object coordinate system $O(x,y,z,w)_k$ and are parametrized by the same parametric coordinate system $P(u,v,w)_j$. The output of the processor is a list of reconstructed control points or boundary curves of parametric patches. This list is passed to the surface editor for conversion of the surface information to surface patches, and hierarchical structuring to surface quadtrees. The individual commands of this command processor are listed in Appendix D.1.

5.2.2 Processor SUED

SUED is the surface editor. Its main task is to convert a list of unstructured patch control points or boundary curves into a structured quadtree of patches, bounding volumes and normal vectors, which can be used by the surface-alignment process. In addition, other surface representations (planar, quadric as well as bicubic) can be entered by SUED commands into its data base. These representations can later be used as 3D shape primitives in the alignment process. Various surface attributes can be specified for each surface element although only reflection and transmission coefficients, and texture functions are useful at the present time. SUED generates a structured surface file which can be used either in surface alignment or in generation of synthetic images. The commands of this processor are described in Appendix D.2.

5.2.3 Processor ALIGN

ALIGN is a processor which performs the 3D alignment of surface descriptions, using the matching and merging algorithms given in Chapter 4. Two surface descriptions are spatially registered with algorithm (4.21). If the two surfaces are segments of an object they are merged into a single parameter coordinate system following the spatial registration. After the two surface descriptions have been matched, and possibly merged, they are returned back to SUED as a linear list of surface patches. SUED then restructures them into a single hierarchical description. ALIGN also performs all the merging algorithms of Section 4.4: parametric surfaces are merged by transformation of parameters; algebraic and parametric surfaces are parametrized or reparametrized by parameter projections; and surfaces of a solid object are converted to a volume representation made of parallelepipeds. A description of the individual commands of this processor may be found in Appendix D.3.

5.2.4 Processor STRAW

STRAW is a processor which paints pretty pictures in highly non-linear time. It renders images of complex 3D scenes with surface-to-surface reflections, transparent surfaces with refractions, diffuse and specular reflections, image and texture surface mappings, illumination by point-light sources and shadows. STRAW uses a ray-tracing algorithm to compute visible surfaces and a recursive shading algorithm, developed by Whitted [72], to compute light intensity of the visible surfaces.

A raster image computed by this program is considered to be an array of square or rectangular pixels. The 3D scene is sampled at the four corner points of each pixel. In perspective projection a 3D sample ray from an image sample point to the center of projection is extended into the scene and intersected with the nearest surface element. In orthographic projection each 3D sample ray is perpendicular to the image plane. The intensity of the sample point is evaluated from visual properties of the intersected surface element as well as from intensity information provided by additional rays which are bounced from the intersection point in the reflection, refraction, and light-source directions. Information about each reflected and refracted ray is maintained in a node of a binary shading tree. Finally, the intensity of a pixel is computed by averaging its four sample values. Anti-aliasing is performed by recursively subdividing a pixel, which has a large difference in its sampled values, into 2 x 2 regions and repeating the sampling process at the four corners of each new region. The intensity of each region is again the average of its four samples and the final intensity of the pixel is the sum of all regional intensities, each weighted by its area. This subdivision is done only within pixels which contain sharp intensity changes, typically caused by edges, silhouettes or textures.

STRAW operates on the structured surface descriptions provided by SUED. The ambient intensity, i.e., the color of a surface element can be specified as a constant value [Figure 5.1(a)], the parameters _u_, _v_ of the element can be used to look up the value in a color image [Figure 5.1(c,d)], or the value can be computed by linear interpolation of colors along 3D vectors stored in a paint table [Figure 5.1(b)]. A texture can be added to a surface element with Blinn's texturing technique [12]. This technique perturbs the normal vector to a surface point in a direction specified by the partial derivatives of a 2D texture function. Here, the source of the texture function is a monochrome image and the parameters _u_, _v_ of a surface element are used to look up values in the image from which the partial derivatives are computed. This texturing is quite effective; however, it remains only a shading illusion - the silhouettes of sur-

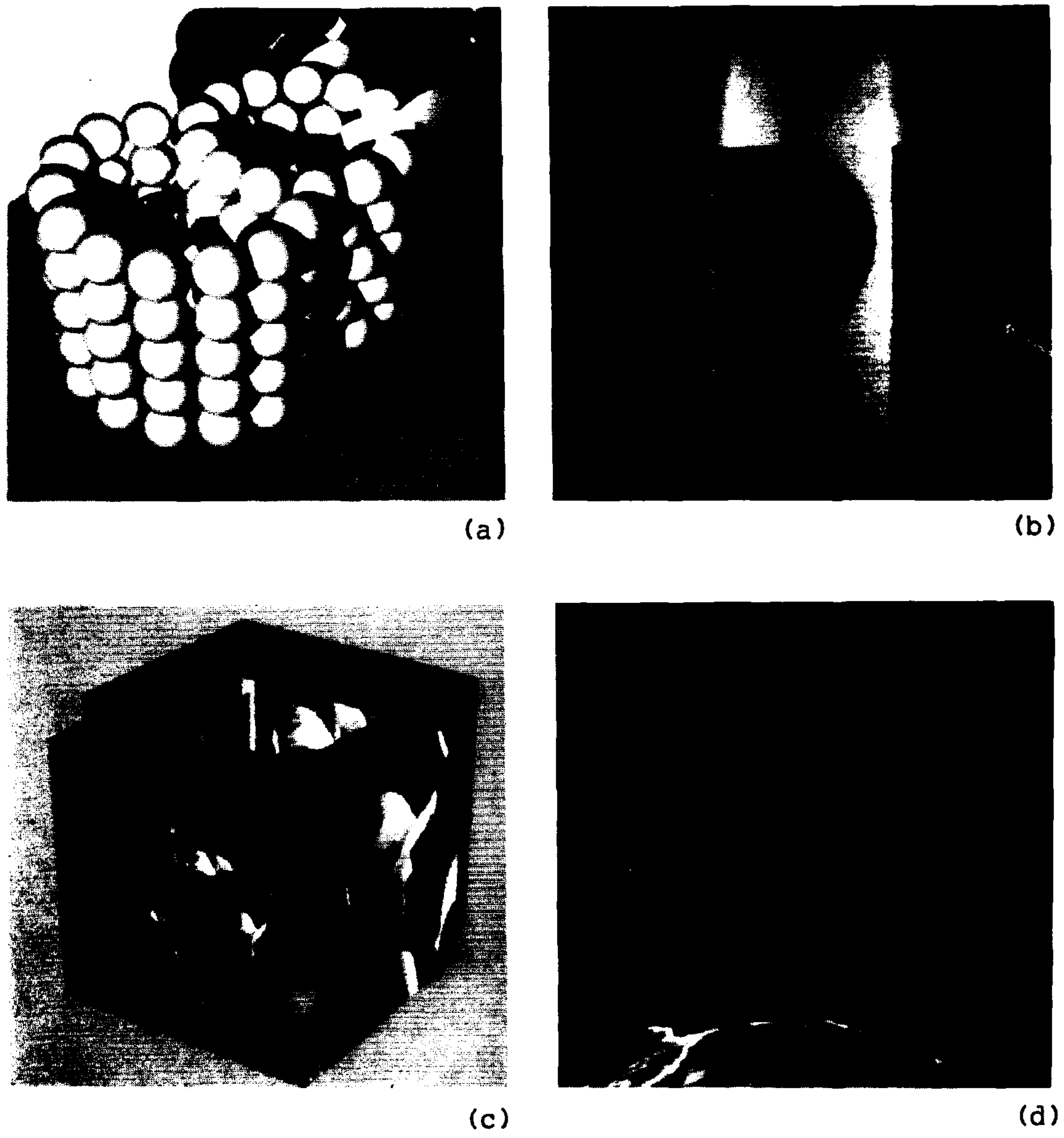
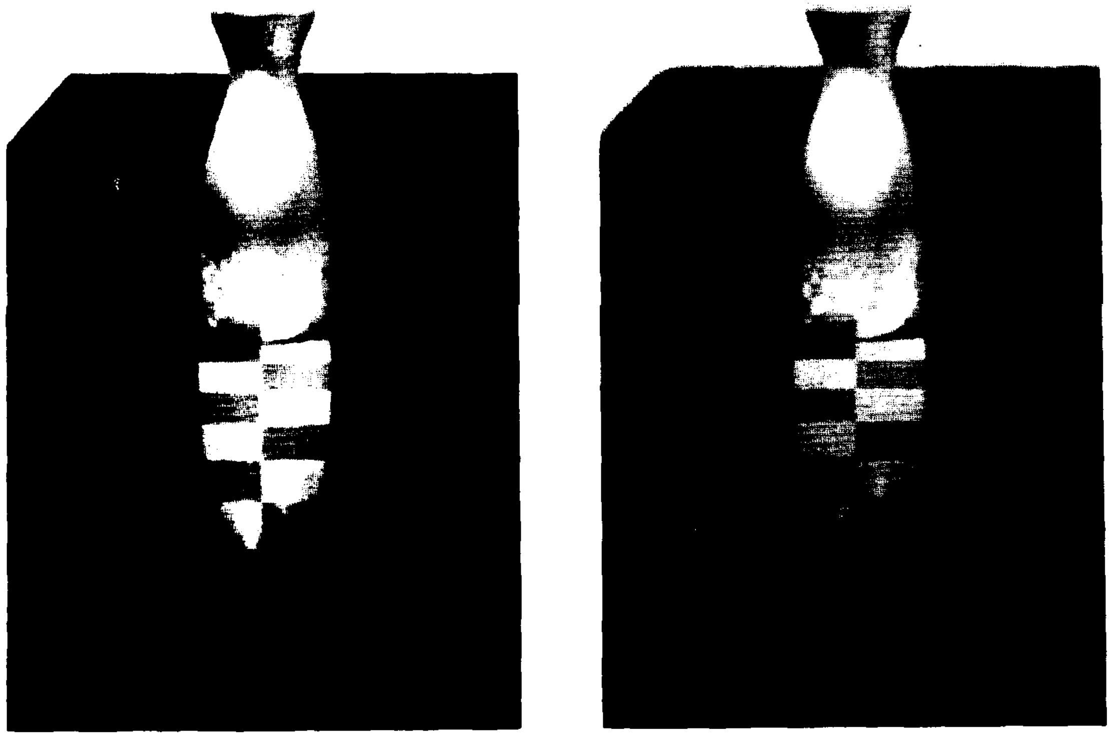


Figure 5.1 Examples of synthetic images generated by STRAW: (a) Microtubular doublet, (b) IPL logo, (c) Recursive box, (d) Galaxy far far away ...

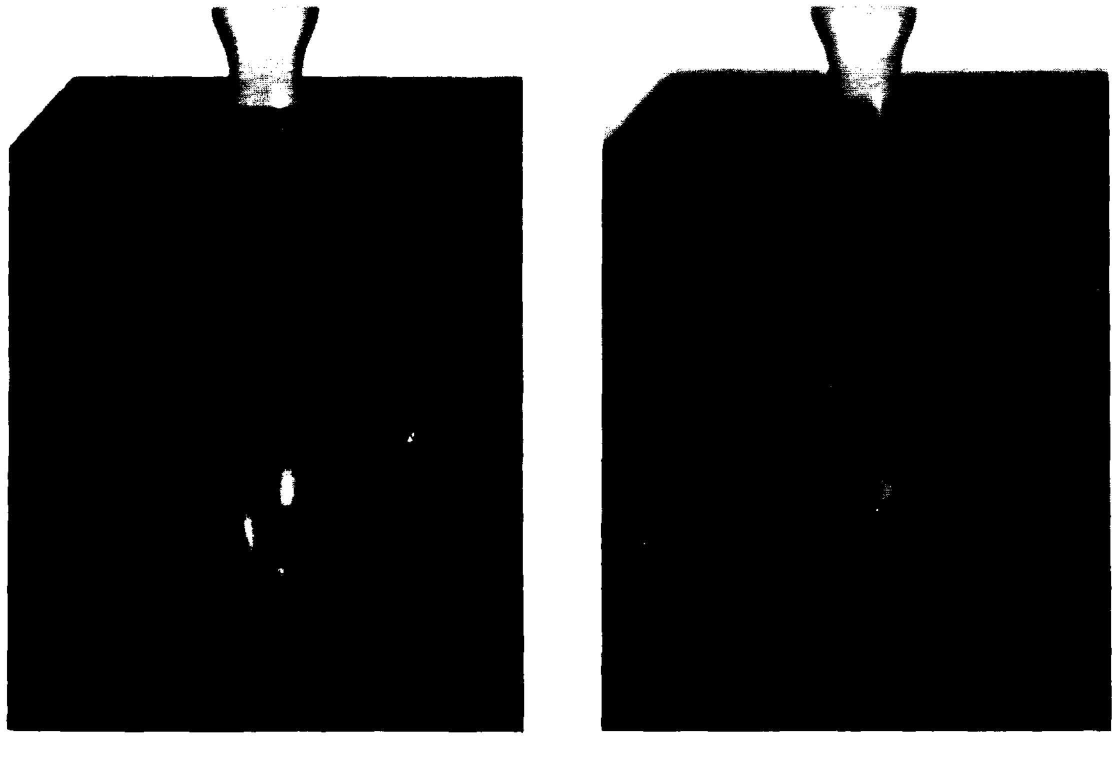
faces are still smooth and the textures cannot cast shadows.

In addition to generating a raster image with a pin-hole camera model, STRAW can optionally save information about each image sample point from the nodes of its shading tree. These sample points may be later converted into an actual raster image by a post-processor. A post-processor, FOCUS, was designed to implement the focus and aperture functions of a camera's lens [57,58]. It generates synthetic images which are focused and have a depth of field [Figures 5.2-3]. For each image sample point, this post-processor computes a point-spread function whose size and intensity distribution depend on the sample's depth (i.e., the distance of the visible surface along the camera's optical axis), the focus distance of the lens, and the aperture size. The point-spread function is then convolved with the sample point in the spatial domain. Since FOCUS has available information about each ray in each shading tree, it can properly defocus reflected [Figure 5.2] and refracted [Figure 5.3] rays of light. A post-processor, BLUR, adds motion blur [59], due to a finite exposure time of real cameras, to moving surfaces [Figure 5.4]. This is accomplished by convolving all image samples which belong to a moving surface with a point-spread function computed from the path and velocity of the motion, and the duration of the exposure. This convolution can be performed equally well in the spatial or the frequency domain.



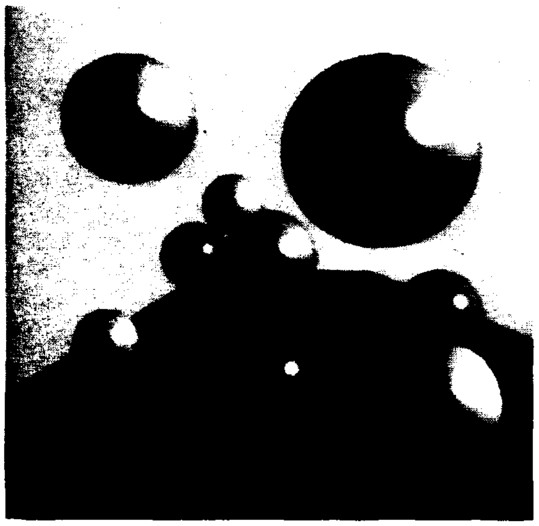
(a) (b)

Figure 5.2 Opaque carafe (a) made with pin-hole camera model, (b) focused on the carafe and aperture set to f/1.4

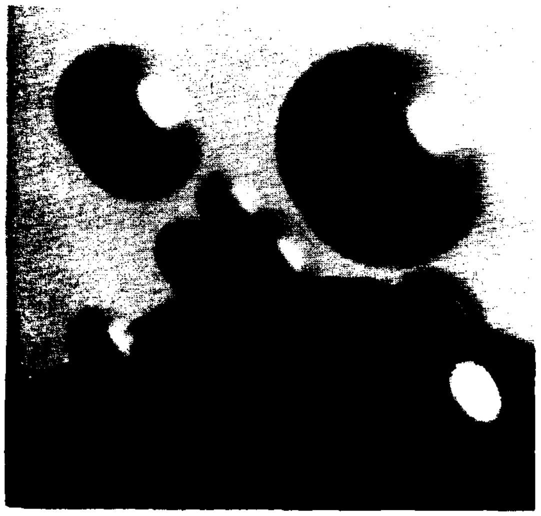


(a) (b)

Figure 5.3 Transparent carafe (a) made with pin-hole camera model, (b) focused on the carafe and aperture set to f/1.4



(a)



(b)

Figure 5.4 Spheres falling on a parabolic path: (a) made with pin-hole camera model, (b) made with motion-blur camera model

Solid objects can be converted by STRAW to the parallelepiped representation (Section 4.4.3) with algorithm (4.40). The line segments inside a solid object of each ray, cast by the orthographic projection, are determined by the ray-tracing traversal procedure. The surface ambient intensity, and the surface-normal vector, oriented to the outside of the object, are evaluated at both ends of each line segment. The ambient intensity may be a constant, image, or paint-table value; the surface-normal vector may be perturbed by a texture function. This description of solid objects is passed to an octree modeling system [44], which converts it to octree representation for efficient analysis and display.

STRAW has also been successfully used to generate stereo pairs of images that can be viewed as transparencies in a stereo viewer. Command STEREO, given a vantage point, a view point and a stereo separation angle, computes the complementary vantage point. Any part of the STRAW data base may be modified between the generation of consecutive image frames that may be used in an animated sequence. This allows, for example, movement of the camera, lights, and surfaces as well as modifications of surface shapes and visual properties. The individual commands of this software processor are listed in Appendix D.4.

CHAPTER 6

EXPERIMENTAL RESULTS

This chapter contains, in the first section, an example of building a complete computer model of a 3D object by the photogrammetric reconstruction and the match-and-merge alignment obtained from several stable positions of the object. This example exercises the entire modeling process of this work. The second section shows two symmetric objects that were partially reconstructed by the photogrammetric procedure, and then completed into generalized cylinder-like shapes with à priori knowledge of their symmetry. Each of these two objects is modeled as a single sheet of surface patches. All the examples given in this chapter contain multi-curved surfaces.

6.1 Car Model

The first example demonstrates the entire modeling process of obtaining a 3D description of an object. It is a balsa model of a car meant for the late 1980's. The computer model was computed from 36 views of the 3D object. The object was positioned in six different object coordinate systems $O(x,y,z,w)_1$ to $O(x,y,z,w)_6$. In each object coordinate system $O(x,y,z,w)_k$, it was consecutively illuminated by three parametric coordinate systems $P(u,v,w)_{1,k}$ to $P(u,v,w)_{3,k}$. While illuminated by a parametric coordinate system $P(u,v,w)_{j,k}$, two images $Q(r,s,w)_{1,j,k}$ and $Q(r,s,w)_{2,j,k}$ were recorded. The tree structure of this modeling process is given in Figure 6.1 with $\underline{I}$ = 36, $\underline{J}$ = 18, and $\underline{K}$ = 6. Four low-resolution (256 x 256 pixel) digital images of the original object are shown in Figure 6.2. A proof (contact) print of the 36 images is given in Figure 6.3. Each column contains six images of the object in one stable position; each pair of images within a row contains the object parametrized by the same parameter system. From the 36 images, there were 18 3D surface segments reconstructed [Figure 6.4]. Since every three segments were defined in the same object coordinate system, they were simply merged into the same parametric coordinate system [Figure 6.5]. In this figure, each drawing in the left column shows three surface segments in $O(x,y,z,w)_k$ before they were merged, and the corresponding drawing in the right column shows the merged surface segment. Algorithm (4.37) was used to reparametrize the surface segments with orthographic projection and 5.0 mm sampling distance. At last, the six merged surface segments were aligned into a common object coordinate system. This alignment process consisted of five iterations; the coordinate system of the first surface segment remained stationary while the other five segments, one at a time, were matched with the existing surface description [Figure 6.6]. Four

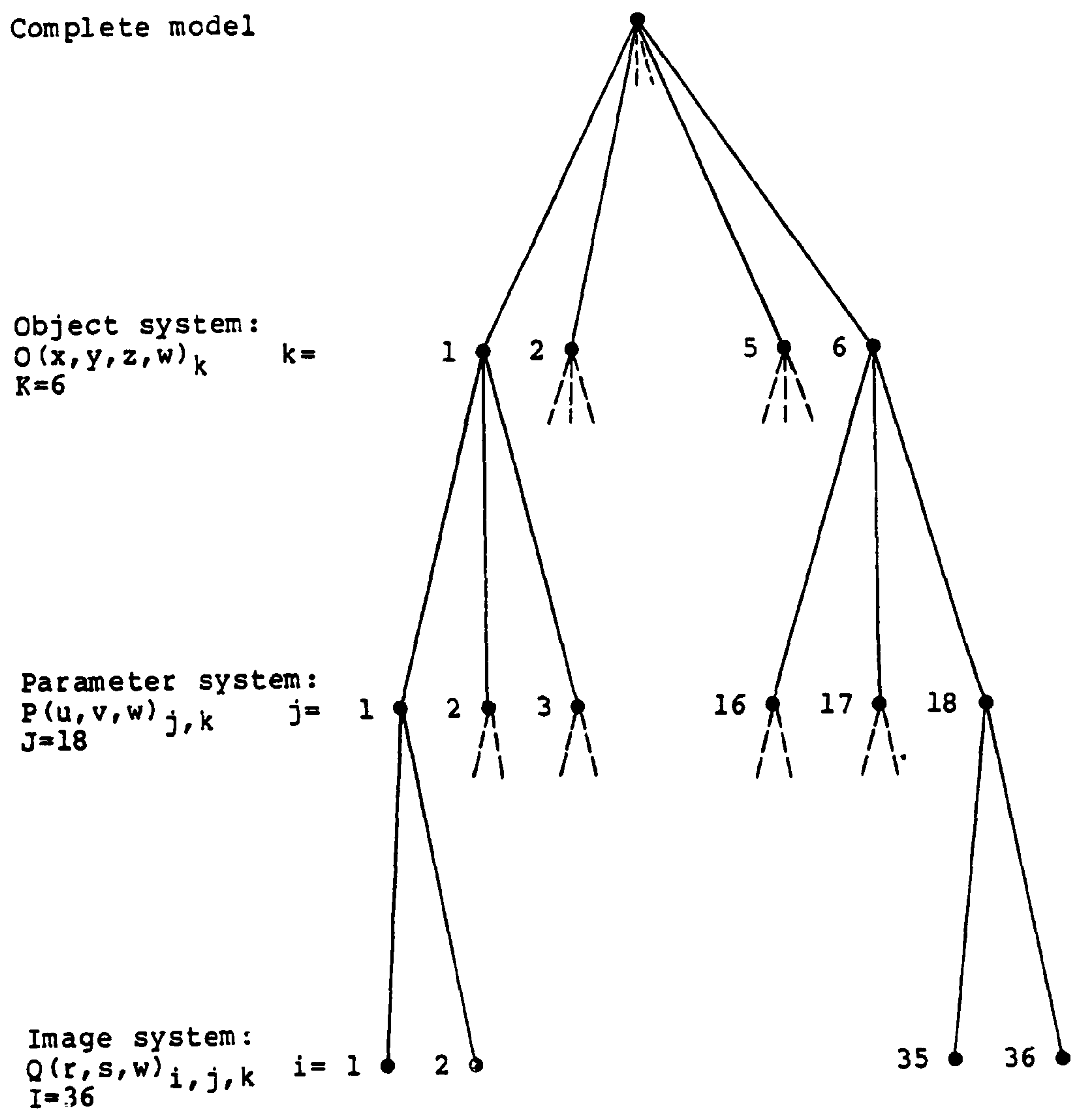


Figure 6.1 Modeling tree for the car model example

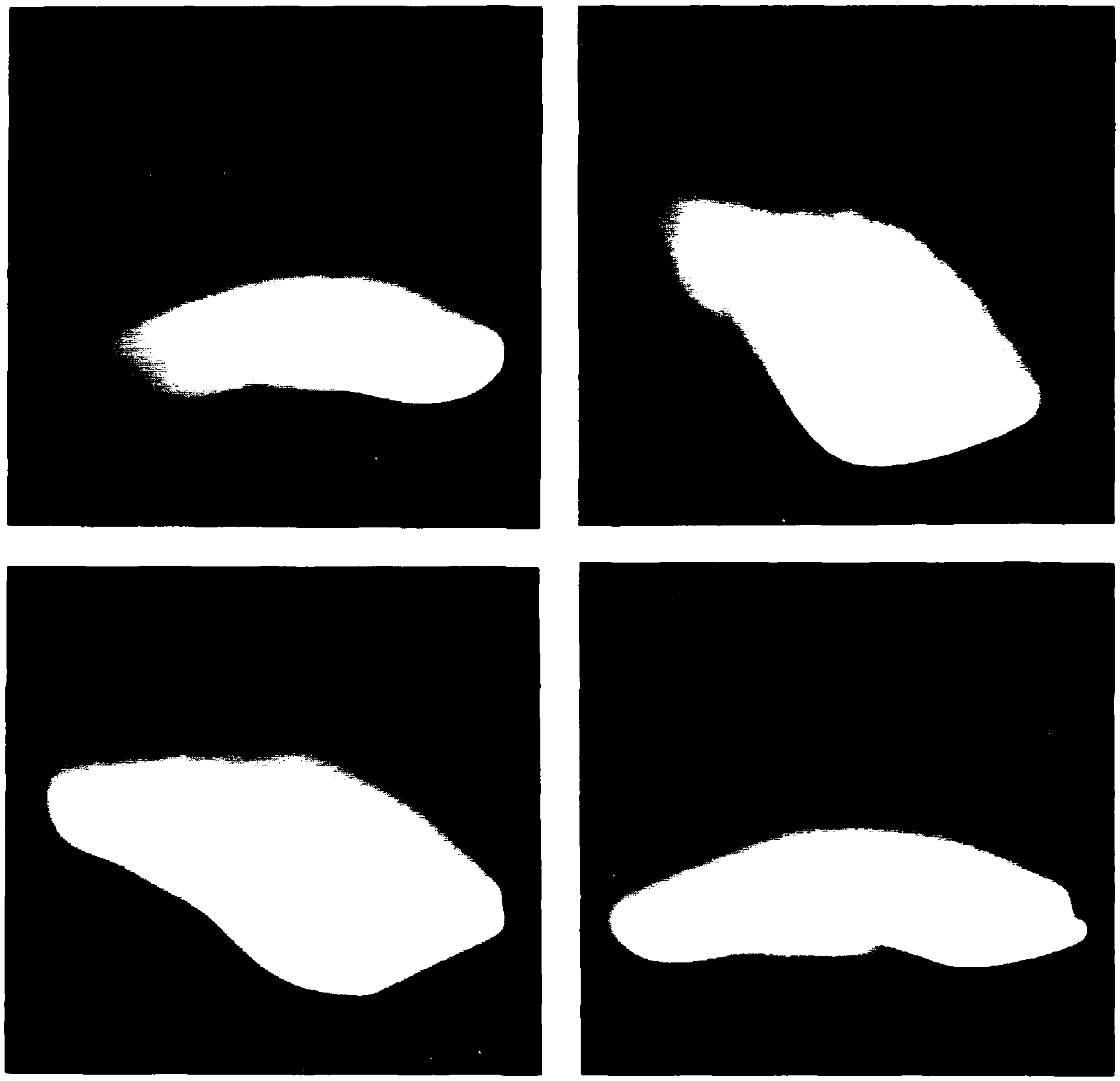


Figure 6.2   Four views of the modeled object: car model
(digital 256 x 256 pixel images)

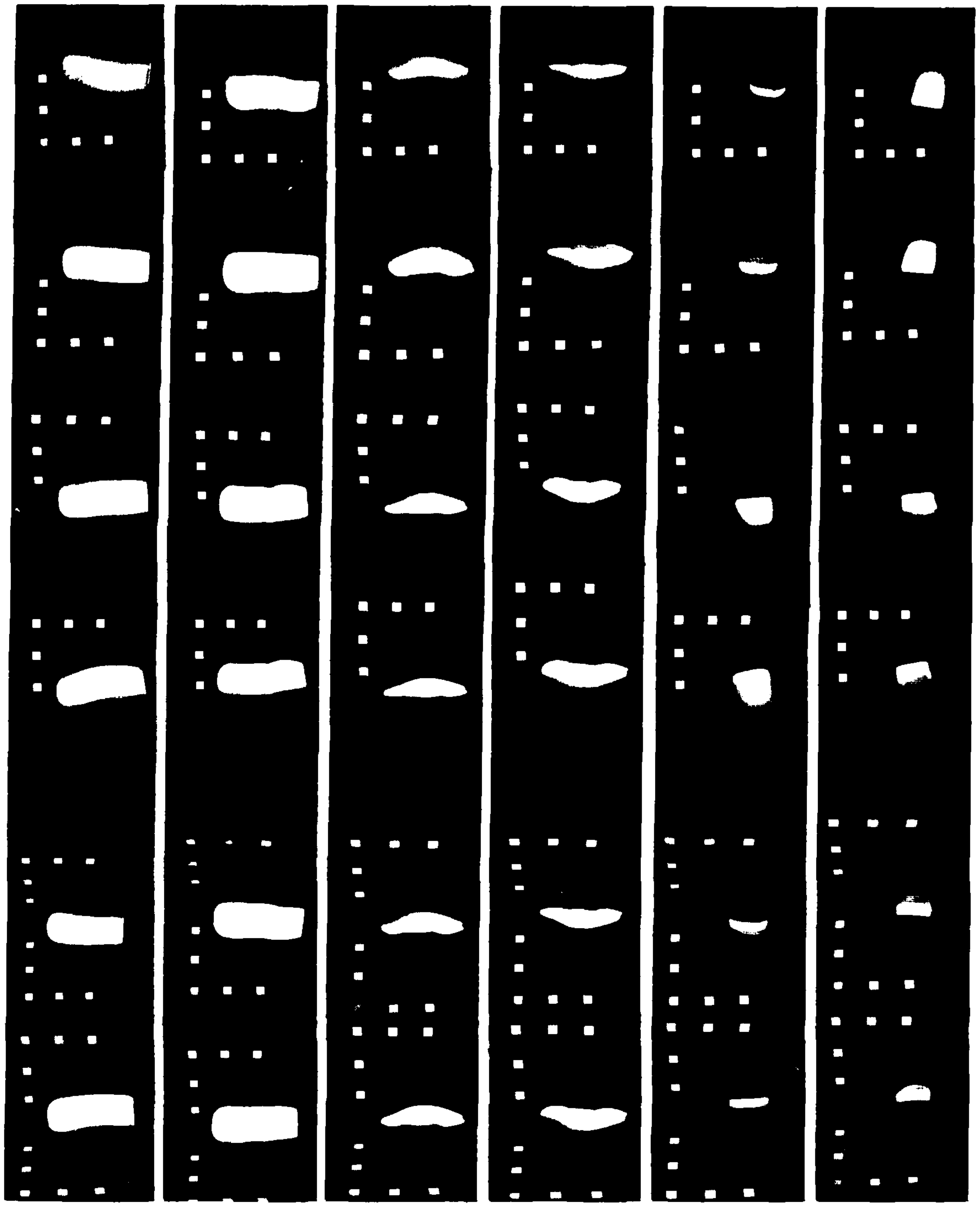


Figure 6.3 Proof print of 36 measured images of the car model

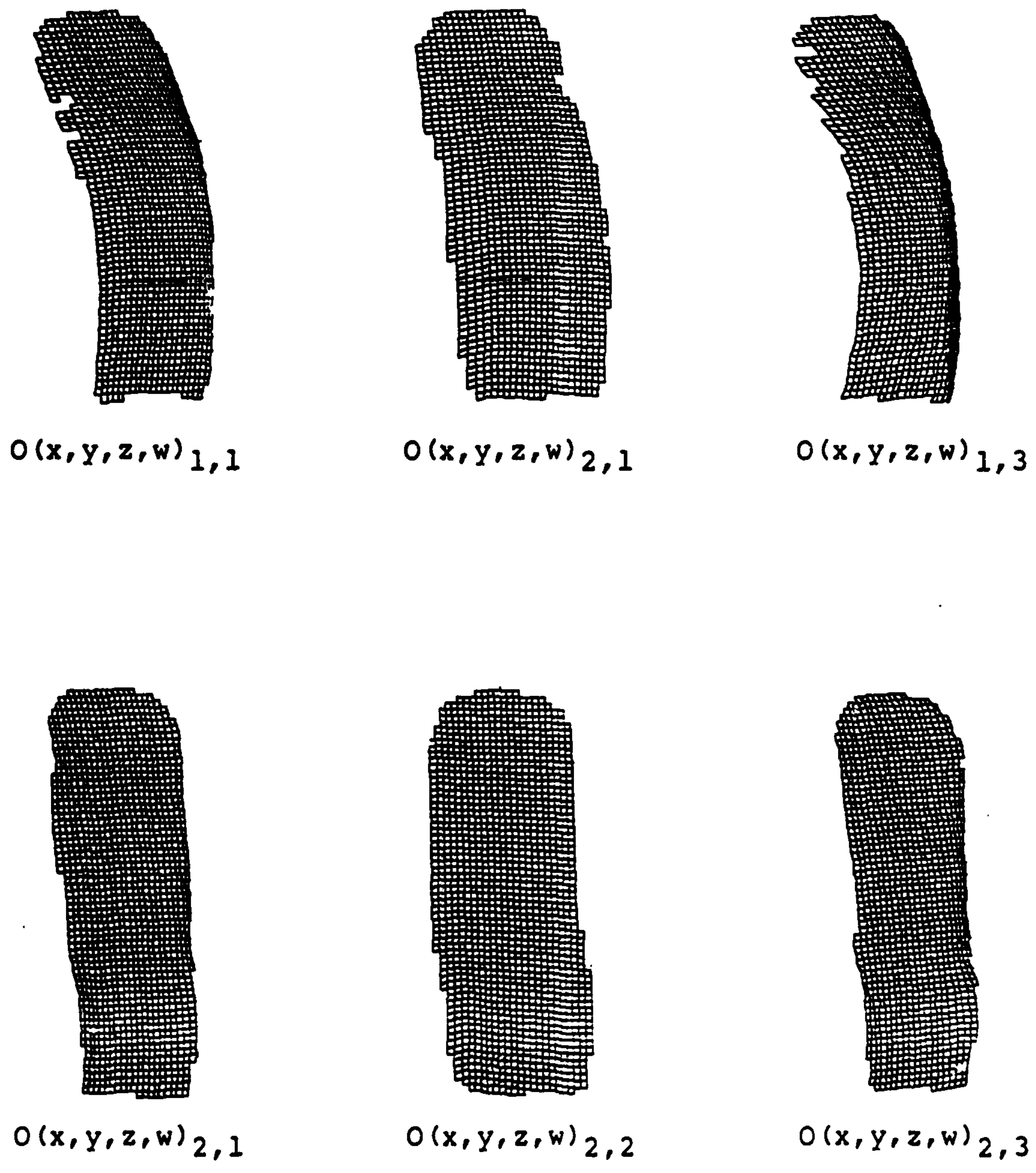
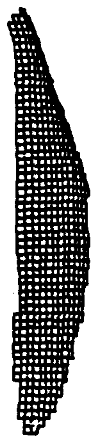
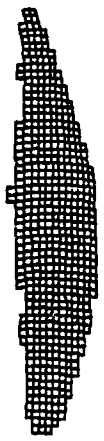


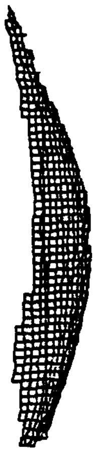
Figure 6.4 Drawings of 18 reconstructed 3D surface segments



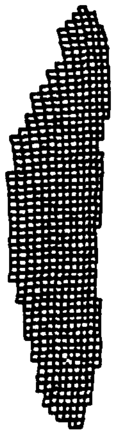
$O(x,y,z,w)_{3,1}$



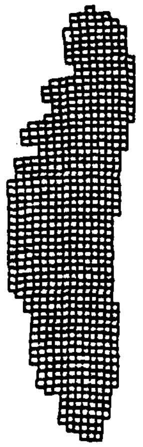
$O(x,y,z,w)_{3,2}$



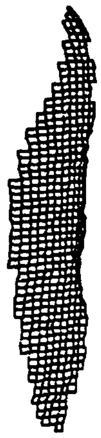
$O(x,y,z,w)_{3,3}$



$O(x,y,z,w)_{4,1}$

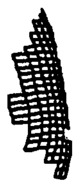


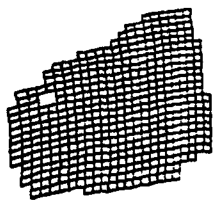
$O(x,y,z,w)_{4,2}$



$O(x,y,z,w)_{4,3}$

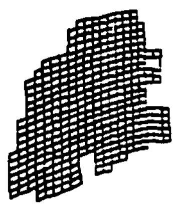
Figure 6.4 (CONTINUED)


$O(x,y,z,w)_{5,1}$


$O(x,y,z,w)_{5,2}$

$O(x,y,z,w)_{5,3}$

$O(x,y,z,w)_{6,1}$


$O(x,y,z,w)_{6,2}$

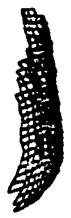
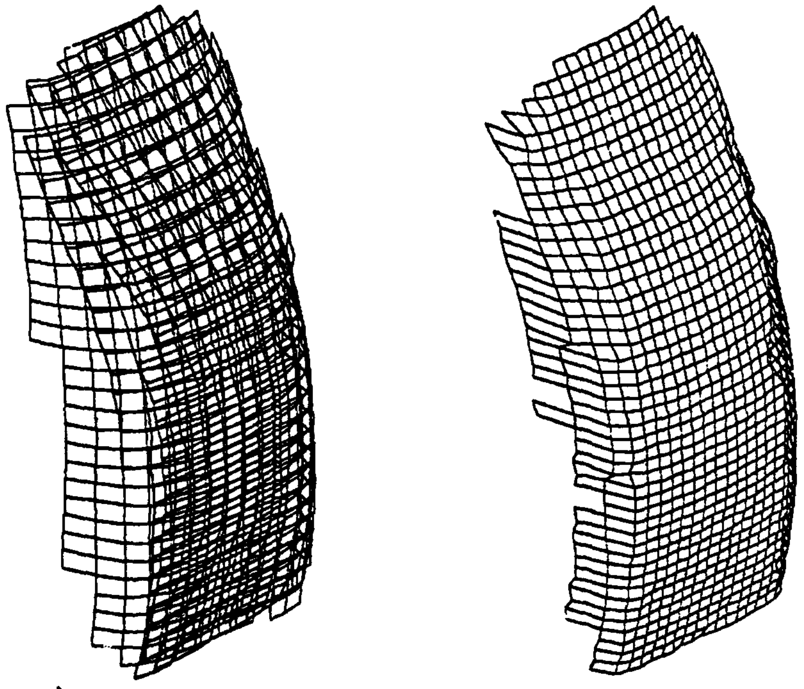

$O(x,y,z,w)_{6,3}$

Figure 6.4 (CONTINUED)


$O(x,y,z,w)_1$

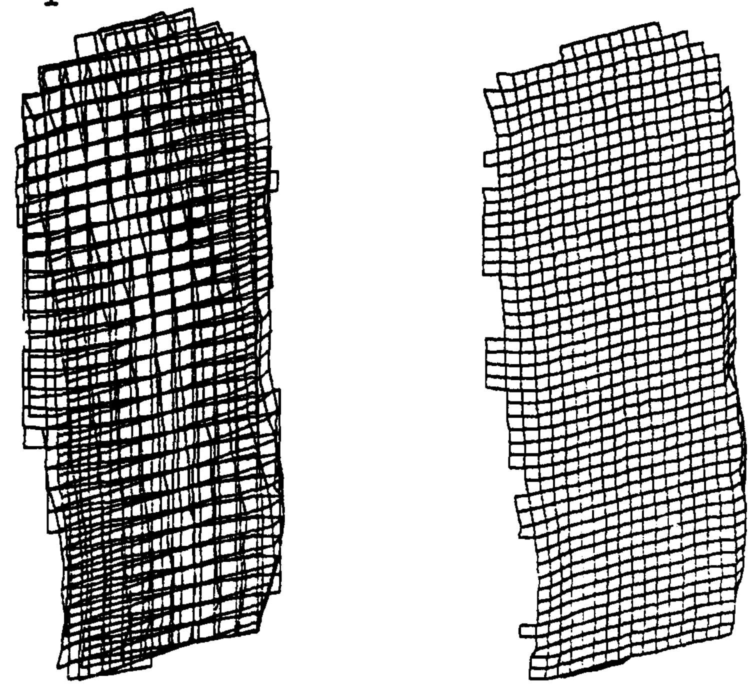
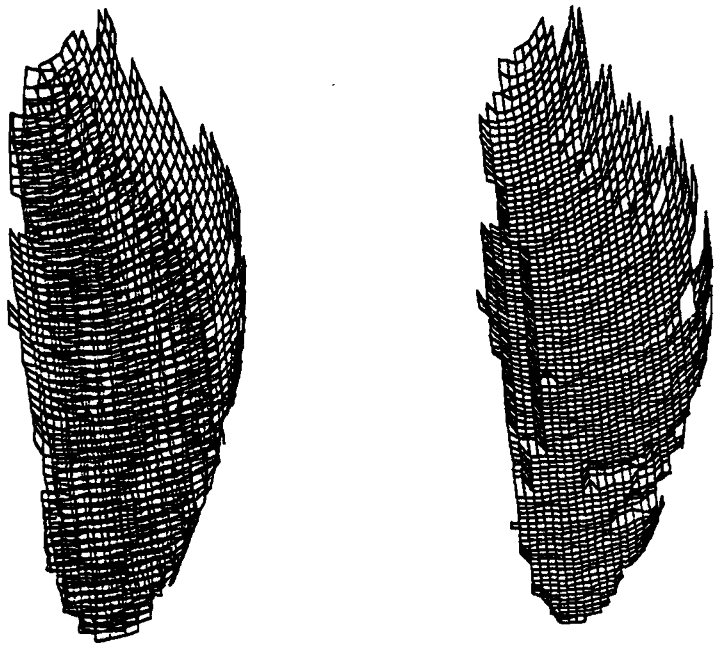
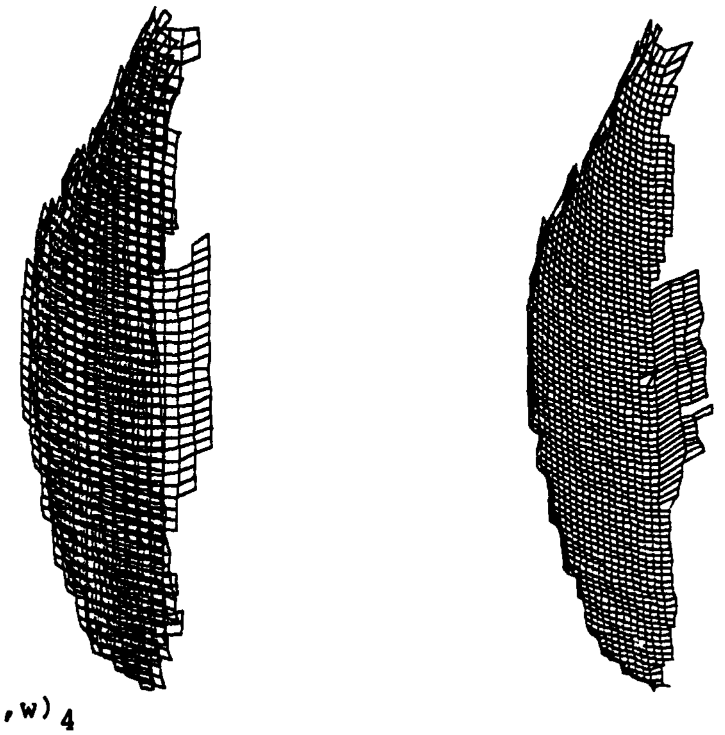

$O(x,y,z,w)_2$

Figure 6.5 Surface segments of Figure 6.4 merged into six segments

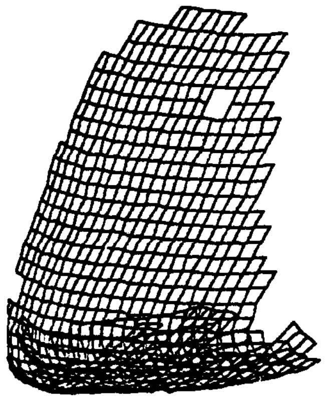
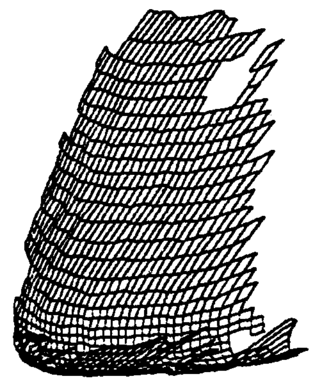


$O(x,y,z,w)_3$

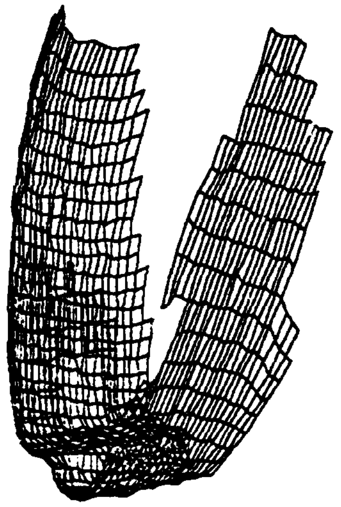
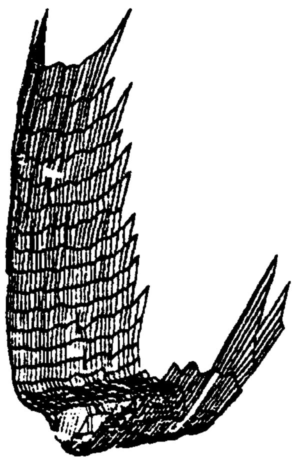


$O(x,y,z,w)_4$

Figure 6.5 (CONTINUED)

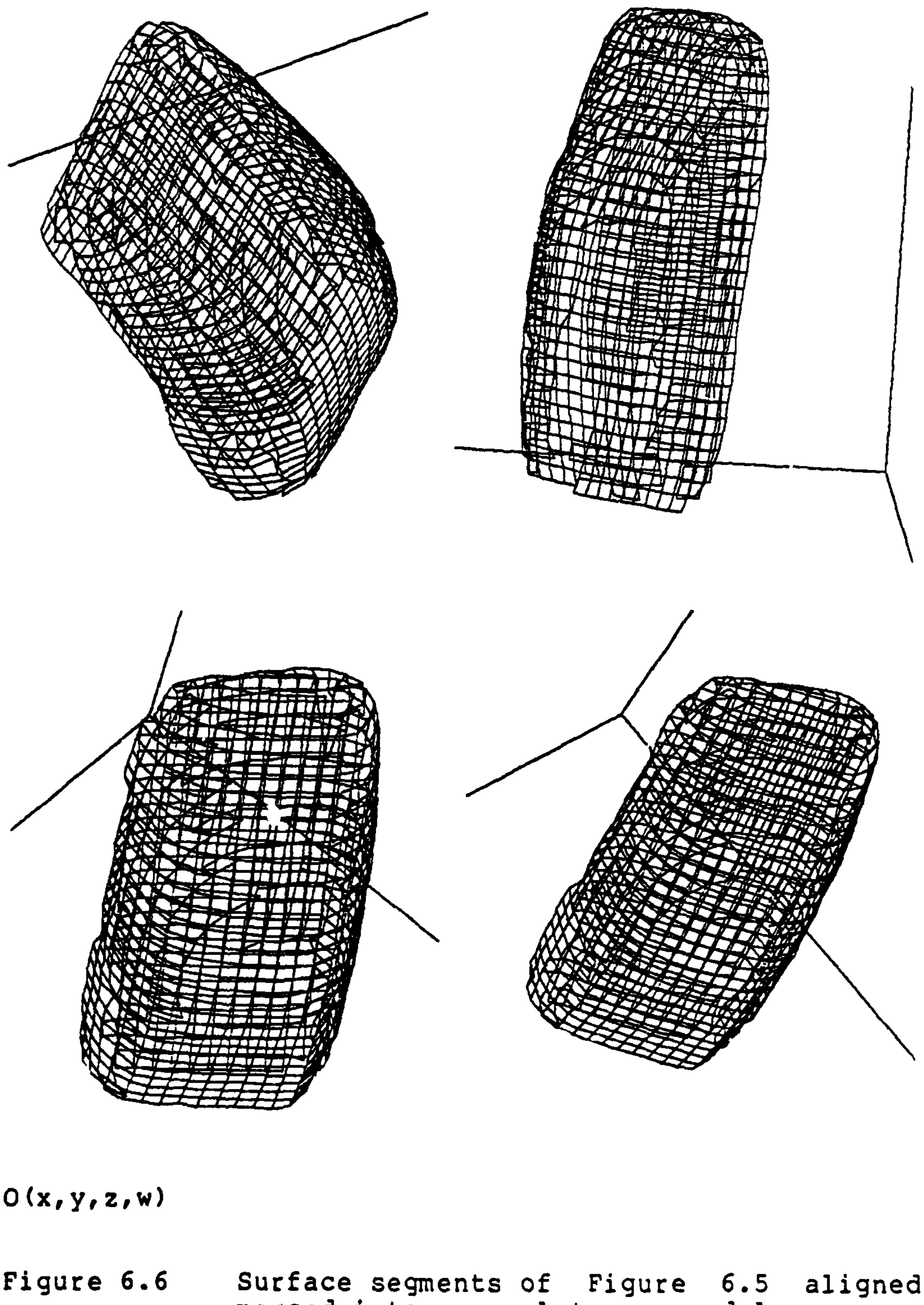



O(x,y,z,w)$_5$

O(x,y,z,w)$_6$

Figure 6.5 (CONTINUED)



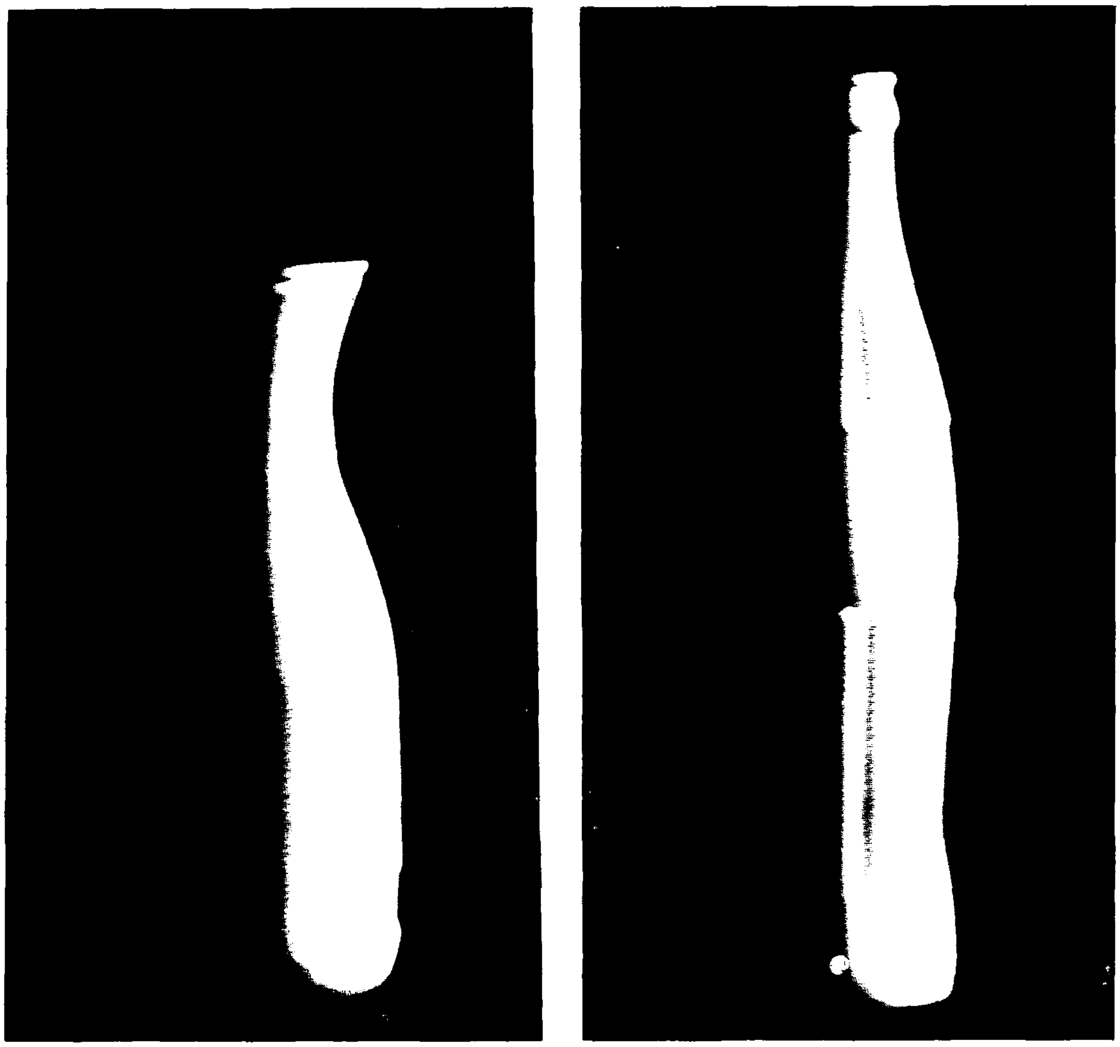
O(x,y,z,w)

Figure 6.6 Surface segments of Figure 6.5 aligned and merged into a complete car model

views of the aligned surface segments are shown in this figure together with the axes of the object coordinate system of the first segment. A single surface model of the object was therefore obtained, and structured into a hierarchical description.

6.2 Symmetric Bottles

This example illustrates the reconstruction of two objects which are symmetric around an axis. Each of the objects - a wine carafe and a coke bottle - was measured in a single object coordinate system $O(x,y,z,w)_1$, while parametrized with a single $P(u,v,w)_{1,1}$ parametric coordinate system, and photographed in two image coordinate systems: $Q(r,s,w)_{1,1,1}$ and $Q(r,s,w)_{2,1,1}$. The section of each object, reconstructed from the two images, was then completed into a generalized cylinder-like shape made of a single sheet of bicubic patches. It was assumed that each object had a circular cross-section in x-y planes. Intersection curves of the reconstructed 3D section with x-y planes equally spaced between the ground plane and the top of the object were computed and circles were fitted into the curves with a least-squares error method. Figure 6.7 shows medium-resolution (512 x 256 pixel) digital images of the original objects. Figures 6.8-6.11 show the results of this modeling procedure. Figures 6.8 and 6.10 contain the two

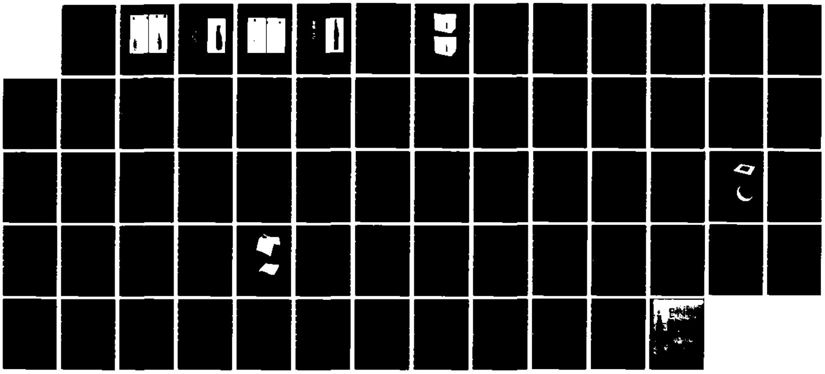


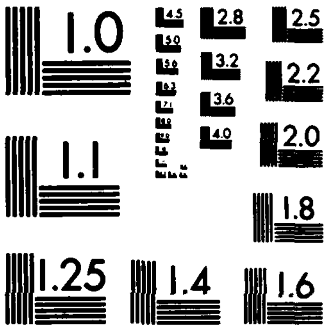
(a) (b)

Figure 6.7 Modeled objects: (a) carafe, (b) bottle (digital 512 x 256 pixel images)

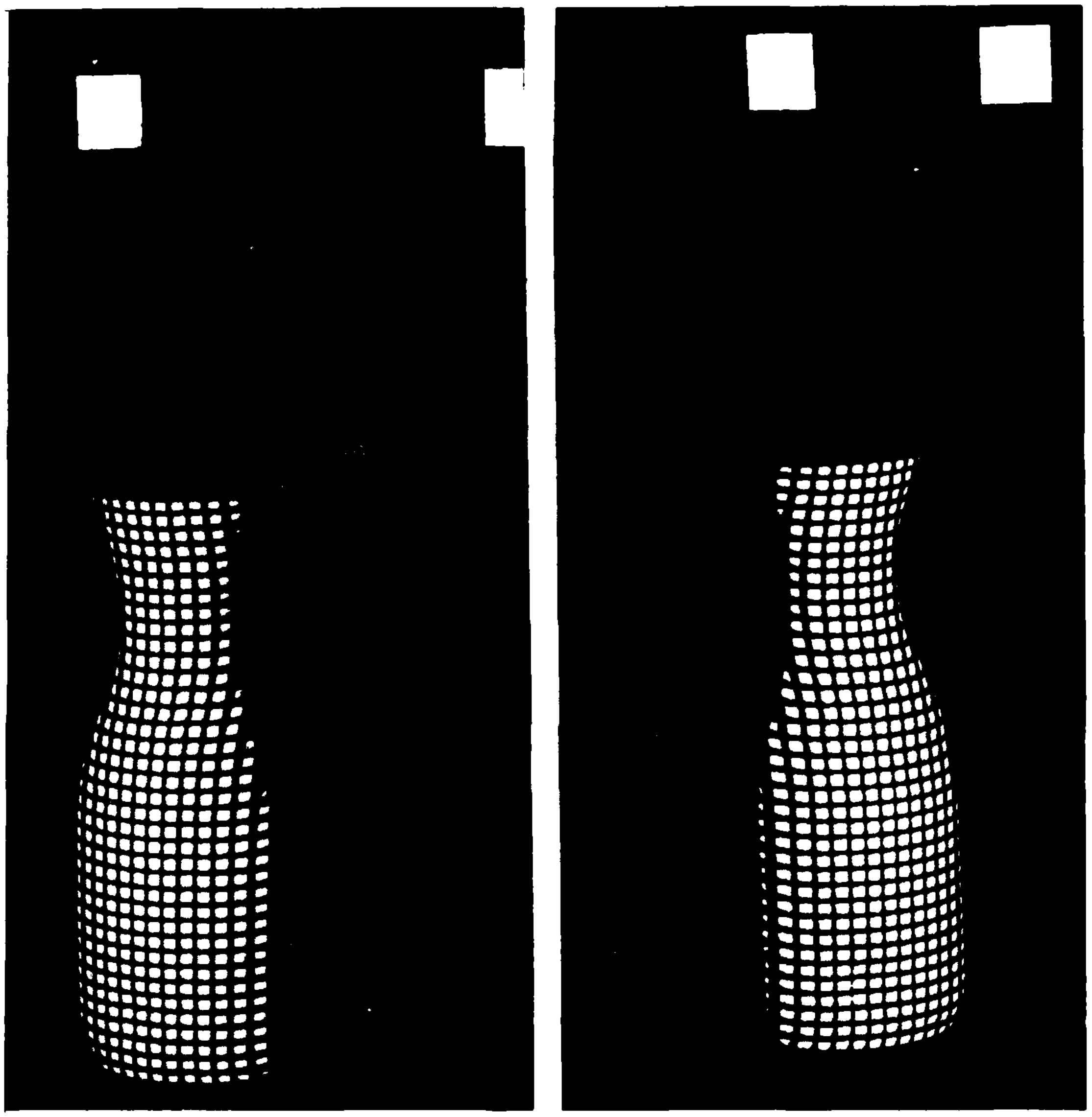
AD-A132 392 GENERATING THREE-DIMENSIONAL SURFACE MODELS OF SOLID OBJECTS FROM MULTIPL..(U) RENSSELAER POLYTECHNIC INST TROY NY IMAGE PROCESSING LAB M POTMESIL OCT 82 3/3

UNCLASSIFIED IPL-TR-033 N00014-82-K-0301 F/G 9/2 NL



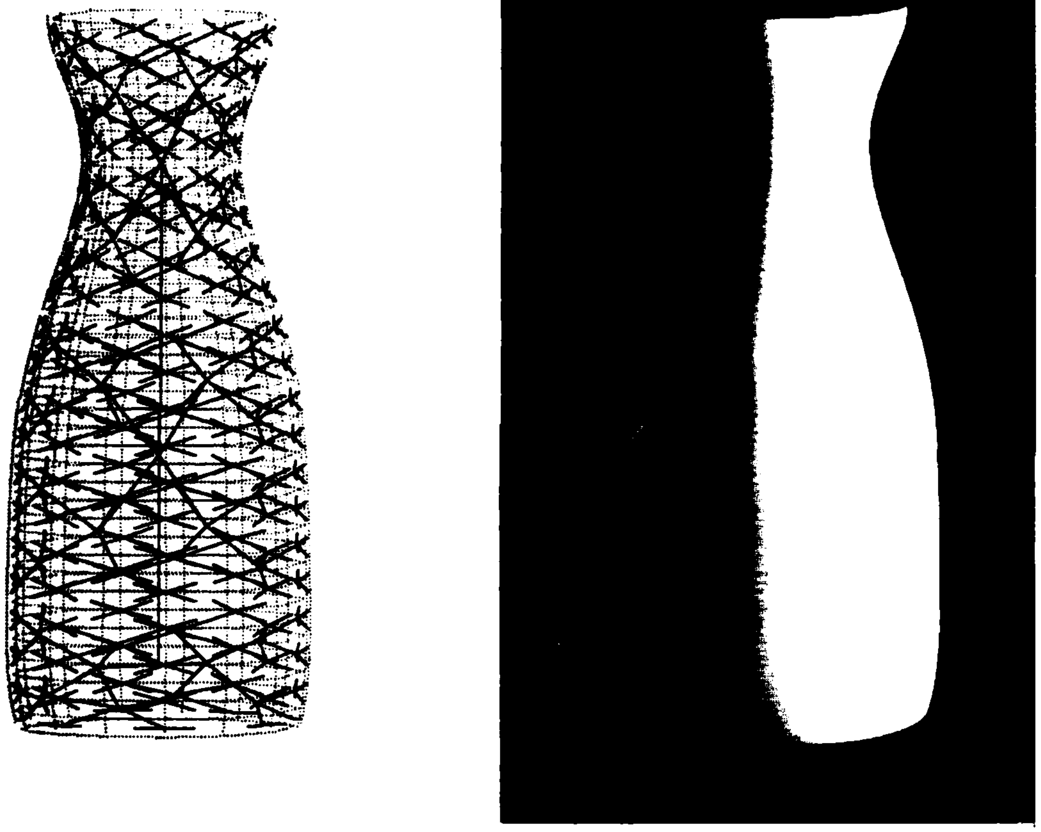


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



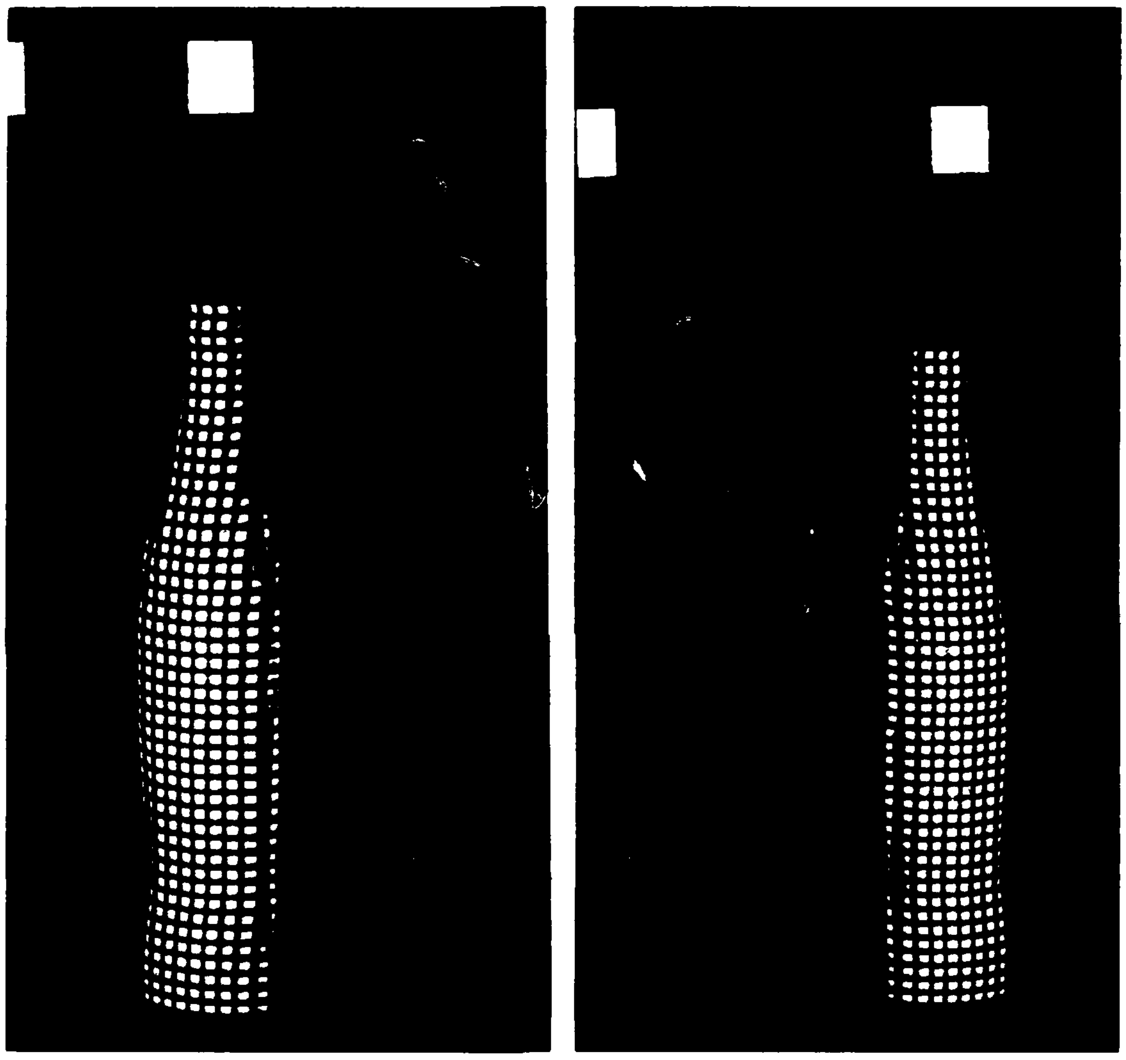
(a) (b)

Figure 6.8 Two images of a parametrized carafe surface: (a) $Q(r,s,w)_{1,1,1}$, (b) $Q(r,s,w)_{2,1,1}$



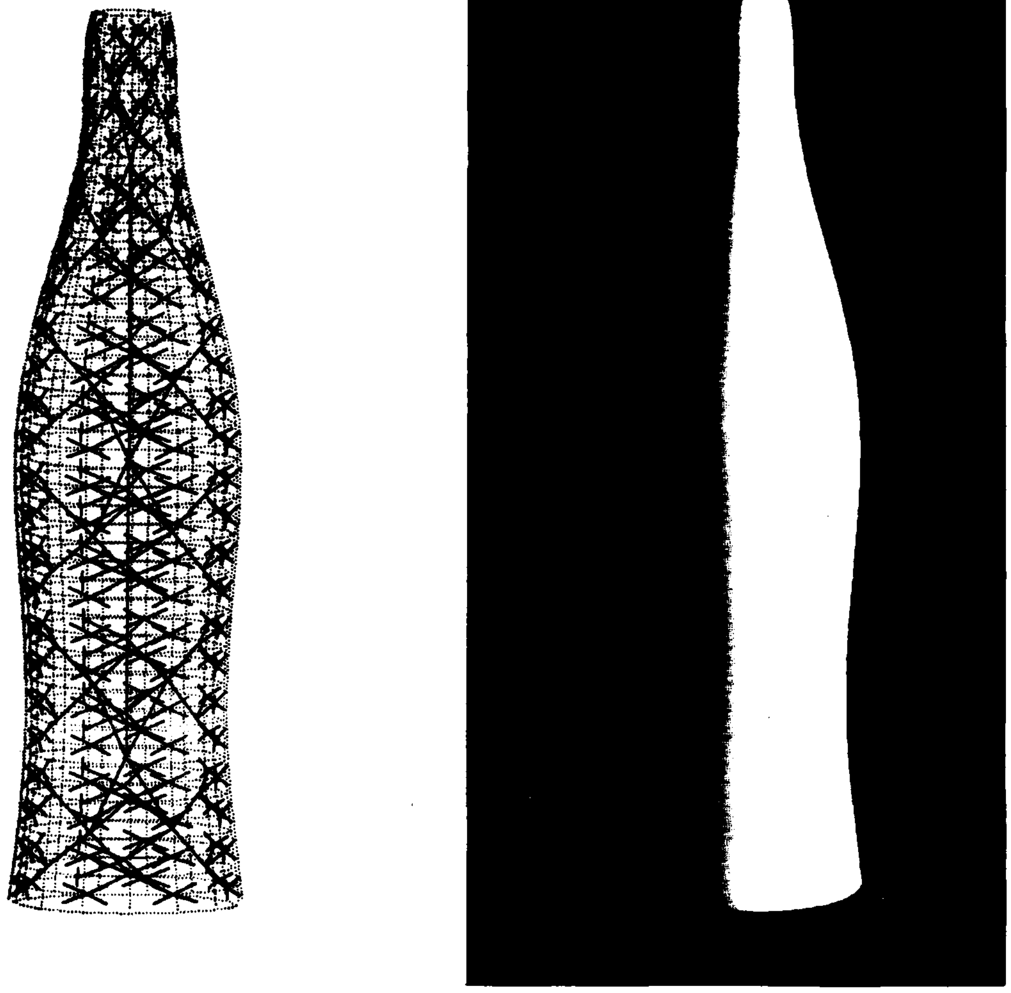
(a) (b)

Figure 6.9 Carafe model: (a) quadtree of surface patches, (b) synthetic image of surface patches



(a) (b)

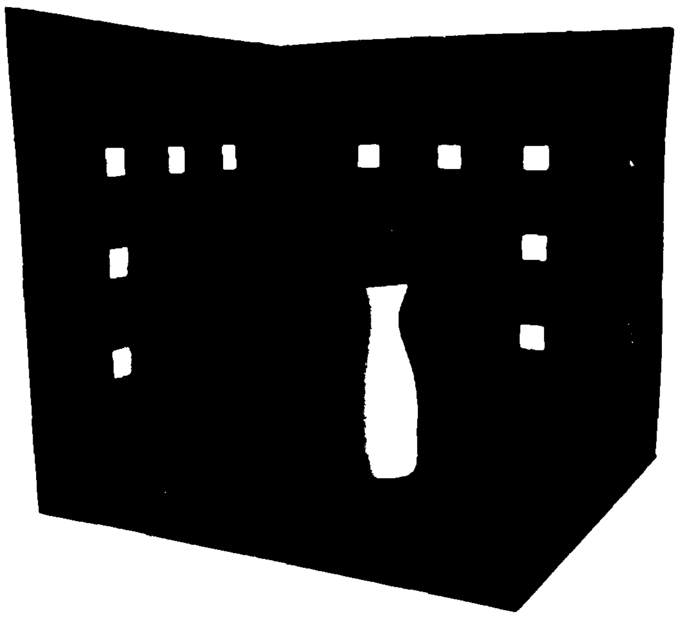
Figure 6.10 Two images of a parametrized bottle surface: (a) $Q(r,s,w)_{1,1,1}$, (b) $Q(r,s,w)_{2,1,1}$



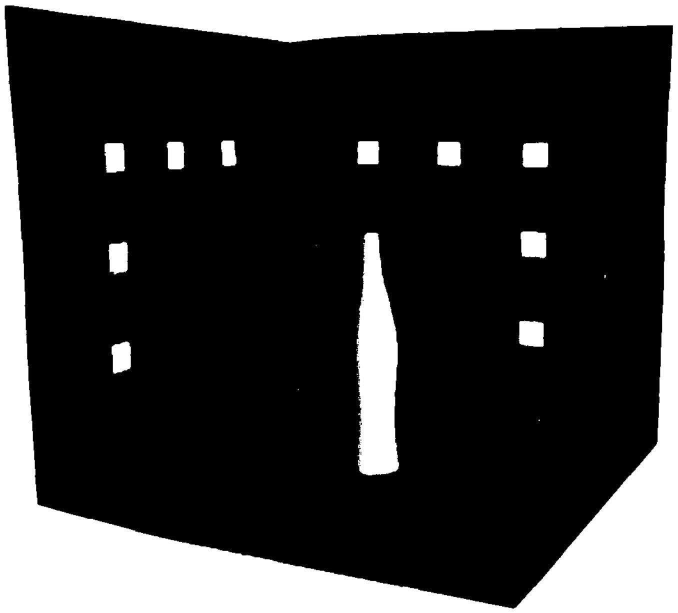
(a) (b)

Figure 6.11 Bottle model: (a) quadtree of surface patches, (b) synthetic image of surface patches

original images of thc parametrized objects. Figures 6.9(a) and 6.11(a) are line drawings of the quadtree representations of the two objects and Figures 6.9(b) and 6.11(b) are shaded synthetic images of the two models. The point-light source in each of these images is placed approximately in the same location at the lamp illuminating the original objects in Figure 6.7. Notice (or rather do not notice) the loss of detail in the surface shape visible along the silhouette of each object. This is partially caused by the resolution of the projected parameteric lines, and also due to using circular cross-sections in the horizontal direction and cubic spline in the the vertical direction of the objects. Figure 6.12 contains two images of these models placed in a model of the camera calibration stand. The model of the carafe is also pictured in Figures 5.2, and 5.3.



(a)



(b)

Figure 6.12 Synthetic images of the objects placed in a model of the camera calibration stand: (a) wine carafe, (b) coke bottle

CHAPTER 7

RETROSPECTS AND PROSPECTS

This chapter summarizes the presented work, lists its major contributions and then proposes tasks for further research.

7.1 Recapitulation

The main objective of this research was to investigate techniques for computer generation and processing of three-dimensional surface models of existing physical objects. The obtained numerical models were to be reasonably precise, complete, and hierarchicaly structured. Models of man-made objects, applicable to both computer vision and CAD/CAM, were of especial interest.

An image processing technique, which digitizes 3D surfaces in a controlled environment (position and illumination of the surfaces), was developed to obtain 3D surface data located on a 2D parametric grid. These data, computed originally at a high resolution, are structured to provide hierarchical surface descriptions. Depending upon the specific amount of surface detail required, various levels of the hierarchical structure are employed. A matching algorithm uses these hierarchical representations to perform

3D alignment of 3D surfaces which share common (overlapping) sections.

A modeling process which assembles complete models of arbitrarily-shaped 3D objects was developed from these capabilities. Individual 3D descriprions of the surfaces of the object are first obtained for different positions of a parameter coordinate system and also for different position of the object in its object coordinate system. The descriptions, given in the same object system but different parameter systems, are merged into a common parameter system. Those descriptions, that are given in different object systems, are first spatially matched into a common object system, and then merged into a common parameter system. If 3D data covering all the surfaces of an object are available, this process can iteratively build a single complete model. The model representation developed here is versitile enough to be directly used in a number of applications; it can also be easily expanded and converted to other representations. Two immediate applications are (a) conversion to octree representation for fast object analysis and display [44], and (b) generation of characteristic views for object recognition [17]. A single ray-tracing procedure provides all required information about a surface description to all application programs. All geometric operations which depend on surface representation are confined to this procedure.

The surface and object models can have assigned reflection, transmission, and texture characteristics, and be viewed in computer-generated raster images. The circle of computer graphics and image processing is then completed by applying image processing techniques to these synthetic images. The inverse process of "image restoration," used in image processing to remove noise from natural images, is applied here to add noise, caused by the optical system of the simulated camera, to the computer graphics images.

7.2 Contributions

The following may be considered as the major accomplishments of this work:

(1) Development of a 3D surface-data acquisition system based on spatially-controlled illumination of the measured surfaces in multiple photographic images; reconstruction of surface information from 2D perspective projections into a 3D representation.

(2) Hierarchical structuring of the 3D surface representation into surface quadtrees of varying levels of detail.

(3) An algorithm for computing intersections of a 3D line and a 3D bicubic parametric patch.

(4) Alignment of 3D surface descriptions based on surface shape using heuristic search methods. Generation of numerical models of objects from aligned

and merged surface descriptions.

(5) Design and implementation of a software system to accomplish the above items and demonstration of its performance using several modeling examples.

(6) Design and implementation of a software system for generation of synthetic images of the modeled objects; expansion of the pin-hole camera model traditionally used for generation of synthetic images to include several parameters of an actual optical system such as focus, depth of field, and motion blur.

7.3 Suggestions

The following is a list of suggestions for possible continuation and expansion of the work presented here:

(1) Segmentation of 3D numerical object models into 3D surface shape and volumetric primitives and top-down structuring of these primitives.

(2) Matching of surface segments which do not overlap, only share boundary curves, i.e., fragments of broken pottery.

(3) Detection of 3D surface intersections and 3D surface vertices (points where three or more surfaces intersect) using 3D curvature search.

(4) Development of relational matching technique of 3D

objects based on the data supplied by the modeling process described here. Automatic conversion of numerical representation into symbolic relations and symbolic surface description.

(5) Expansion of the optical model of a camera's lens to include special-effect filters (such as star an diffraction), and noise caused by an optical sys tem.

REFERENCES

1. Agin, G. J., "Hierarchical Representation of Three-Dimensional Objects," Final Report, SRI Project 1187, Stanford Research Institute, Palo Alto, California, March 1977

2. Agin, G. J., and Binford, T. O., "Computer Description of Curved Objects," Proc. 3IJCAI, Stanford, California, August 1973, 629-640

3. Badler, N., and Bajcsy, R., "Three-Dimensional Representation for Computer Graphics and Computer Vision," ACM Computer Graphics (Proc. SIGGRAPH '78), 12, (3), August 1978, 153-160

4. Baer, A., Eastman, C., and Hension, M., "Geometric Modeling: A Survey," Computer-Aided Design, 11, (5), September 1977, 253-272

5. Bajcsy, R., "Three-Dimensional Scene Analysis," Proc. 4ICPR, Miami Beach, Florida, December 1980, 1064-1074

6. Baker, H., "Three-Dimensional Modelling," Proc. 5IJCAI, Cambridge, Massachusetts, August 1977, 649-655

7. Barnhill, R. E., and Riesenfeld, R. F. (Eds.), Computer Aided Geometric Design, Academic Press, 1974

8. Barr, A. H., "Superquadrics and Angle-Preserving Transformations," IEEE Computer Graphics and Applications, 1, (1), January 1981

9. Barrow, H. G., Tenenbaum, J. M., Bolles, R. C., and Wolf, H. C., "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching," Proc. 5IJCAI, Cambridge, Massachusetts, August 1977, 659-663

10. Baumgart, B., "Geometric Modeling for Computer Vision," AIM-249, Stanford University, Palo Alto, California, October 1974

11. Bhanu, B., "Shape Matching and Image Segmentation Using Stochastic Labeling,", USCIPI Report 1030, Image Processing Institute, University of Southern California, Los Angeles, California, August 1981

12. Blinn, J. F., "Simulation of Wrinkled Surfaces," ACM Computer Graphics (Proc. SIGGRAPH '78), 12, (3), August 1978, 286-292

13. Burr, D. J., "On Computer Stereo Vision with Wire Frame Models," UILU-ENG 77-2251, University of Illinois at Urbana-Champaign, Urbana, Illinois, December 1977

14. Burr, D. J., "Elastic Matching of Line Drawings," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-3, (6), November 1981, 708-713

15. Catmull, E., "A Subdivision Algorithm for Computer Display of Curved Surfaces," UTEC CSc-74-133, Computer Science Department, University of Utah, Salt Lake City, Utah, December 1974

16. Catmull, E., and Clark, E., "Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes," Computer-Aided Design, 10, (6), November 1978, 350-355

17. Chakravarty, I. and Freeman, H., "Characteristic Views as a Basis for Three-Dimensional Object Recognition," Proc. SPIE Technical Symposium East '82, Arlington, Virginia, May 1982 (to appear)

18. Clark, J. H., "Hierarchical Geometric Models for Visible Surface Algorithms," Comm. ACM, 19, (10), October 1976, 547-554

19. Conte, S. D., and de Boor, C., Elementary Numerical Analysis: An Algorithmic Approach, McGraw-Hill, New York, 1972

20. Coons, S. A., "Surfaces for Computer-Aided Design of Space Forms," M.I.T. Project MAC TR-41, June 1967, (AD-663 504)

21. Davis, L. S., "Shape Matching Using Relaxation Techniques," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1, (1), January 1979, 60-72

22. do Carmo, M. P., Differential Geometry of Curves and Surfaces, Prentice-Hall, New Jersey, 1976

23. Duda, R. O, and Hart, P. E., "Use of the Hough Transformation to Detect Lines and Curves in Pictures," Comm. ACM, 15, (1), January 1972, 11-15

24. Duda, R. O. and Hart, P. E., Pattern Classification and Scene Analysis, John W'ley Interscience, New York, 1973

25. England, J. N., "A System for Interactive Modeling of Physical Curved Surface Objects," ACM Computer Graphics (Proc. SIGGRAPH '78), 12, (3), August 1978, 336-340

26. Faux, I. D., and Pratt, M. J., Computational Geometry for Design and Manufacture, John Wiley and Sons, New York, 1979

27. Ferguson, J. C., "Multivariate Curve Interpolation," Journal ACM, 11, (2), February 1964, 221-228

28. Freeman, H., and Potmesil, M., "Curved Surface Representation Utilizing Data Extracted from Multiple Photographic Images," Proceedings of a Workshop on the Representation of Three-Dimensional Objects, University of Pennsylvania, May 1979, H1-H26

29. Freeman, H., and Chakravarty, I., "The Use of Characteristic Views in the Recognition of Three-Dimensional Objects," Pattern Recognition in Practice (Gelsema, E., and Kanal, L., eds.), North-Holland Publishing Company, Amsterdam, 1980

30. Fuchs, H., Kedem, Z. M., and Uselton, S. P., "Optimal Surface Reconstruction from Planar Contours," Comm. ACM, 20, (10), October 1977, 693-702

31. Fuchs, H., Kedem, Z. M., and Naylor B. F., "On Visible Surface Genaration by A Priory Tree Structures," ACM Computer Graphics (Proc. SIGGRAPH '80), 14, (3), July 1980, 124-133

32. Grossman, D. D., and Taylor, R. H., "Interactive Generation of Models with Manipulator," IEEE Transactions on Systems, Man and Cybernetics, 8, (9), September 1978, 667-679

33. Grossman, D. D., "Procedural Representation of Three-Dimensional Objects," IBM J. Res. Dev., 20, November 1976, 582-589

34. Hall, E. L., Computer Image Processing and Recognition, Academic Press, New York, 1979

35. Horn, B. K. P., "Obtaining Shape from Shading Informa-

tion," In Psychology of Computer Vision (Winston, P. H., ed.), McGraw-Hill, 1975

36. Horn, B. K. P., and Bachman, B. L., "Using Synthetic Images with Surface Models," Comm. ACM, 21, (11), November 1978, 914-924

37. Hunter, G. M., and Steiglitz. K., "Operations on Images Using Quad Trees," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1, (2), April 1979, 145-153

38. Idesawa, M., Yatagai, T., and Soma, T., "Scanning Moire Method and Automatic Measurement of 3-D Shapes," Applied Optics, 16, (8), August 1977, 2152-2162

39. Idesawa, M., and Yatagai, T., "3-D Shape Input and Processing by Moire Technique," Proc. 5ICPR, Miami Beach, Florida, December 1980, 1085-1090

40. Lane, J. M., Carpenter, L. C., Whitted, T. and Blinn, J. F., "Scan Line Methods for Displaying Parametrically Defined Surfaces," Comm. ACM, 23, (1), January 1980, 23-34

41. Levin, J. Z., "QUISP: A Computer Processor for the Design and Display of Quadric-Surface Bodies," IPL-TR-80-003, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, New York, April 1980

42. Lozano-Perez, T., and Winston, P., "LAMA: A Language for Automatic Mechanical Assembly," Proc. 5IJCAI, Cambridge, Massachusetts, August 1977, 710-716

43. Martelli, A., "Edge Detection Using Heuristic Search Method," Computer Graphics and Image Processing, 1, (2), August 1973, 162-183

44. Meagher, J. R., "Octree Encoding: A New Technique for The Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer," IPL-TR-80-111, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, New York, October 1980

45. Marshall, R., Wilson, R., and Carlson W., "Procedure Models for Generating Three-Dimensional Terrain," ACM Computer Graphics (Proc. SIGGRAPH '80), 14, (3), July 1980, 154-162

46. Nevatia, R. and Binford, T. O., "Description and

Recognition of Curved Objects," AI, 8, (1), February 1977, 77-98

47. Newell, M. E., "The Utilization of Procedure Models in Digital Image Synthesis," UTEC CSc-76-218, Computer Science Department, University of Utah, Salt Lake City, Utah, 1975

48. Nilsson, N. J., Problem Solving Methods in Artificial Intelligence, McGraw Hill, New York, 1970

49. Nilsson, N. J., Principles of Artificial Intelligence, Tioga Press, Palo Alto, California, 1980

50. O'Rourke, J., and Badler, N., "Decomposition of Three-Dimensional Objects into Spheres," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1, (3), July 1979, 295-305

51. Oshima M., and Shirai, Y., "Representation of Curved Objects Using Three-Dimensional Information," Proc. Second USA-JAPAN Computer Conference, AFIPS Press, 1975, 108-112

52. Pavlidis, T., Algorithms for Graphics and Image Processing, Computer Science Press, Maryland, 1981

53. Posdamer, J. L., and Altschuler, M. D., "Surface Measurement by Space-Encoded Projected Beam Systems," Computer Graphics and Image Processing, 18, (1), January 1982, 1-17

54. Potmesil, M., and Freeman, H., "Computer Descriptions of Curved-Surface Objects from Multiple Projected-Pattern Images," TR CRL-58, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, New York, June 1978

55. Potmesil, M., "DBS - A PR1ME Data Base System," Bulletin U-107, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, New York, May 1979

56. Potmesil, M., "Generation of 3D Surface Descriptions from Images of Pattern-Illuminated Objects," Proceedings of IEEE Computer Society Conference on Pattern Recognition and Image Processing, Chicago, Illinois, August 1979, 553-559

57. Potmesil, M., and Chakravarty, I., "A Lens and Aperture Camera Model for Synthetic Image Generation,"

ACM Computer Graphics (Proc. SIGGRAPH '81), 15, (3), August 1981, 297-305

58. Potmesil, M., and Chakravarty, I., "Synthetic Image Generation with A Lens and Aperture Camera Model," ACM Transactions on Graphics, 1, (2), April 1982 (to appear)

59. Potmesil, M., and Chakravarty, I., "Modeling Motion Blur in Computer-Generated Images," (in preparation)

60. Pratt, W. K., Digital Image Processing, John Wiley Interscience, New York, 1978

61. Rabinowitz, A., "Reconstruction of Polyhedra from Sets of Their Perspective Projections," Ph.D. Thesis, Department of Electrical Engineering, New York University, New York, April 1971, (AD 732 300)

62. Requicha, A. A. G., "Representations for Rigid Solids: Theory, Methods and Systems," ACM Computing Surveys, 12, (4), December 1980, 437-464

63. Roth, S. D., "Ray Casting for Modeling Solids," Computer Graphics and Image Processing, 18, (1), January 1982, 109-144

64. Rubin, S. M., and Whitted, T., "A 3-Dimensional Representation for Fast Rendering of Complex Scenes, ACM Computer Graphics (Proc. SIGGRAPH '80), 14, (3), July 1980, 110-116

65. Samet, H., "An Algorithm for Converting Rasters to Quadtrees," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-3, (1), January 1981, 93-95

66. Schumaker, L. L., "Fitting Surfaces to Scattered Data," In Approximation Theory II, (Lorenz, G. G., Chui, C. K., and Schumaker, L. L., eds.), Academic Press, 1976

67. Shamos, M. I., and Hoey, D., "Closest-Point Problems," Department of Computer Science, Yale University, New Haven, Connecticut 06520

68. Shapira, R., "Computer Reconstruction of Bodies Bounded by Quadric Surfaces from a Set of Imperfect Projections," TR CRL-48, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, New York, September 1976, (AD-A035 563)

69. Shapiro, L. G., Haralick, R. M., Moriarty, J. D., and Mulgaonkar, P. G., "Matching Three-Dimensional Models," Proceedings of IEEE Computer Society Conference on Pattern Recognition and Image Processing, Dallas, Texas, August 1981, 534-541

70. Sobel, I., "On Calibrating Computer Controlled Cameras for Perceiving 3-D Scenes," Proc. 3IJCAI, Stanford, California, August 1973, 648-655

71. Tsukiama, T., "Scene Partition Using Three Dimensional Data," Progress Report on 3-D Object Recognition, Electrotechnical Laboratory, Japan, Match 1977, 123-127

72. Whitted, T., "An Improved Illumination Model for Shaded Display," Comm. ACM, 23, (6), June 1980, 343-349

73. Will, P. M., and Pennington, K. S., "Grid Coding: A Preprocessing Technique for Robot and Machine Vision," AI, 2, (3/4), 1971, 319-329

74. Will, P. M., and Pennington, K. S., "Grid Coding: A Novel Technique for Image Processing," Proc. IEEE, 60, (6), June 1972, 669-680

75. Winston, P. H., "Learning Structural Descriptions from Examples," In The Psychology of Computer Vision (Winston, P. H., ed.), McGraw-Hill, 1975

76. Woetzel, G., "A Fast and Economic Scan-to-Line Conversion Algorithm," ACM Computer Graphics (Proc. SIGGRAPH '78), 12, (3), August 1978, 125-129

APPENDIX A

DEFINITION OF A BICUBIC PARAMETRIC PATCH AND A SHEET OF PATCHES

This appendix discusses two methods of constructing bicubic patches from the surface information provided by the photogrammetric technique of Chapter 2, and the construction of sheets of patches from individual but contiguous patches. The first method interpolates a patch into the positional coordinates of four control points, and achieves first-derivative continuity (C^1) across patch boundaries. The second method constructs a patch from its four boundary curves; its continuity across patch boundaries depends on the continuity of the boundary curves which is also C^1.

A bicubic patch is a cubic polynomial of two variables (parameters) expressed, in matrix form, as:

(A.1)

$$g(u,v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

or, for short,

(A.2)

$$g(u,v) = \overline{U}\ \overline{B}\ \overline{V}^t$$

where the domain of the parameters $\underline{u}$, and $\underline{v}$ is the unit square: $0 \le \underline{u}, \underline{v} \le 1$. In equations (A.1-2) g(u,v) actually represents a component of the vector:

$$\bar{g}(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} \tag{A.3}$$

There are several methods of computing the 16 coefficients of matrix $\bar{B}$ in equation (A.1), such as the Coons, Ferguson, Bezier, Hermite, and B-spline methods [7,20,26,27]. The Ferguson method is useful for fitting a patch or a set of patches into existing surface data points; the Coons method fits patches into boundary curves, whereas the other methods are more useful for interactive design of a new surface. In all these methods, however, $\bar{B}$ has to be always computed from 16 items of information about the surface shape.

A <u>Ferguson patch</u> [27] is constructed from surface data of four control points as follows; let

$$\bar{B} = \bar{M}\,\bar{H}\,\bar{M}^t \tag{A.4}$$

where

$$\bar{M} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{A.5}$$

is a matrix of coefficients of blending functions, and

$$\bar{H} = \begin{bmatrix} h(0,0) & h(0,1) & h_v(0,0) & h_v(0,1) \\ h(1,0) & h(1,1) & h_v(1,0) & h_v(1,1) \\ h_u(0,0) & h_u(0,1) & h_{uv}(0,0) & h_{uv}(0,1) \\ h_u(1,0) & h_u(1,1) & h_{uv}(1,0) & h_{uv}(1,1) \end{bmatrix} \tag{A.6}$$

provides the description of the surface data [Figure A.1(a)] with the following short-hand notation:

$$h_u(u,v) = \frac{dh(u,v)}{du},$$

$$h_v(u,v) = \frac{dh(u,v)}{dv}, \text{ and}$$

$$h_{uv}(u,v) = \frac{d^2h(u,v)}{du\ dv},$$

where $h(u,v)$ is the positional coordinate of a control point, $h_u(u,v)$ is the surface tangent in the direction of the parametric curves $\underline{v}$ = constant, $h_v(u,v)$ is the surface tangent in the direction of the parametric curves $\underline{u}$ = constant, and $h_{uv}(u,v)$ is the surface cross-derivative or twist. The values of $h_u(u,v)$, $h_v(u,v)$, and $h_{uv}(u,v)$ can be estimated by numerical-analysis methods [19] from the positional data $h(u,v)$ and adopted to our requirements as follows:

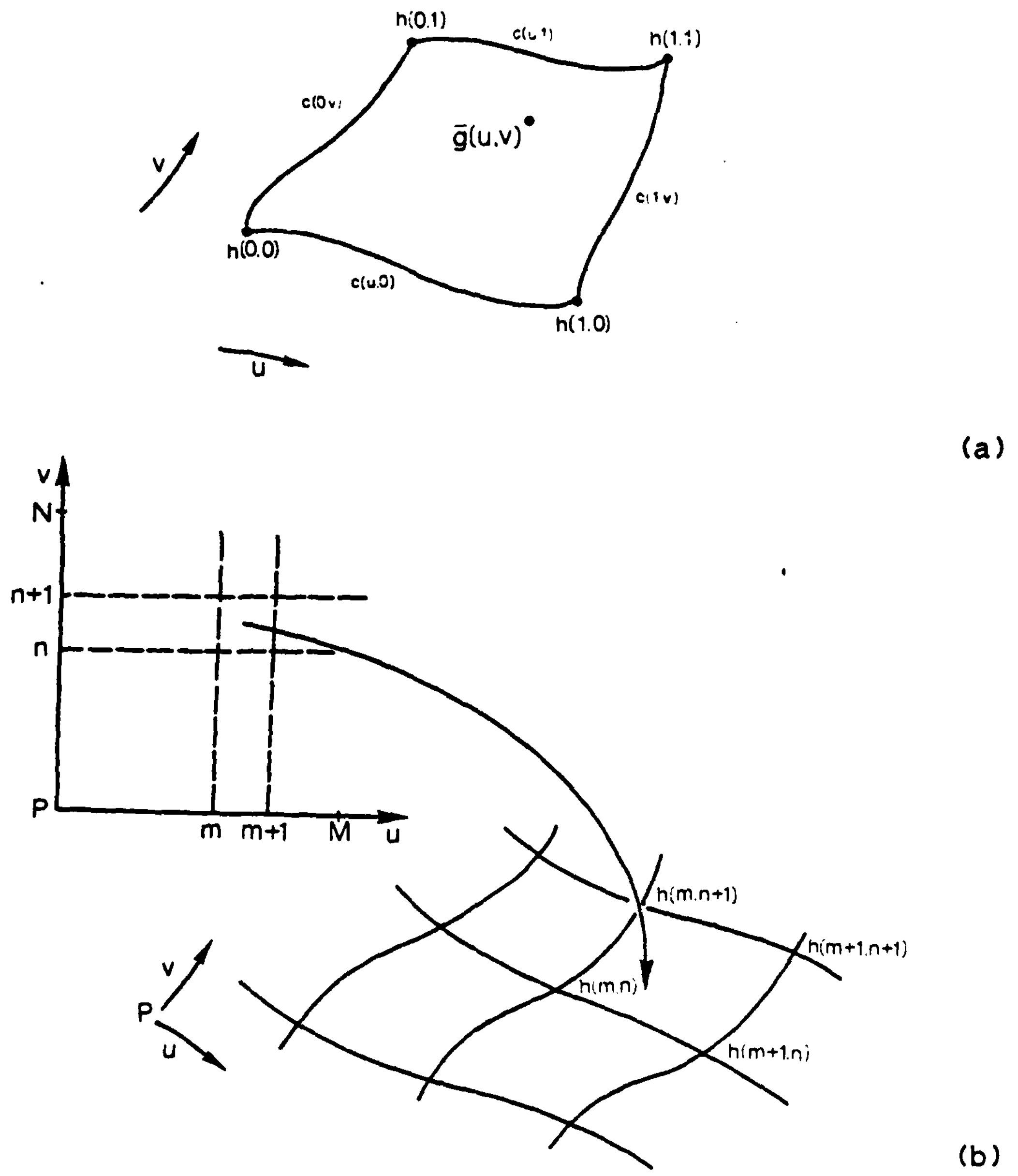


Figure A.1 (a) Bicubic parametric patch, (b) Parametric space and composite parametric patches

$$h_u(u,v) = \frac{h(u+\Delta u,v) - h(u-\Delta u,v)}{2\Delta u} \quad (A.7)$$

$$h_v(u,v) = \frac{h(u,v+\Delta v) - h(u,v-\Delta v)}{2\Delta v} \quad (A.8)$$

$$h_{uv}(u,v) = \frac{[h(u+\Delta u,v+\Delta v)-h(u-\Delta u,v+\Delta v)] - [h(u+\Delta u,v-\Delta v)-h(u-\Delta u,v-\Delta v)]}{4\Delta u\Delta v} \quad (A.9)$$

In equations (A.7-8) above, the surface tangents were computed as central differences; a better estimate of the tangents, yielding a smoother surface, is obtained from a weighted average in a 3 x 3 point neighborhood:

$$h_u(u,v) = \frac{[h(u+\Delta u,v+\Delta v)+2h(u+\Delta u,v)+h(u+\Delta u,v-\Delta v)] - [h(u-\Delta u,v+\Delta v)+2h(u-\Delta u,v)+h(u-\Delta u,v-\Delta v)]}{8\Delta u} \quad (A.10)$$

$$h_v(u,v) = \frac{[h(u+\Delta u,v+\Delta v)+2h(u,v+\Delta v)+h(u-\Delta u,v+\Delta v)] - [h(u+\Delta u,v-\Delta v)+2h(u,v-\Delta v)+h(u-\Delta u,v-\Delta v)]}{8\Delta v} \quad (A.11)$$

In summary, a Ferguson patch is completely defined by [$h(u,v)$ $h_u(u,v)$ $h_v(u,v)$ $h_{uv}(u,v)$] evaluated at each of the four corner control points of the patch and by the blending coefficients of (A.5). For two adjacent patches to have positional and derivative continuities, they must share the surface data defined at their common control points.

A Coons patch [20] is constructed from four boundary curves c(u,0), c(u,1), c(0,v), and c(1,v) [Figure A.1(a)] by linear interpolation between the opposite pairs of boundaries:

$$g(u,v) = \begin{bmatrix} (1-u) & u \end{bmatrix} \begin{bmatrix} c(0,v) \\ c(1,v) \end{bmatrix} + \begin{bmatrix} (1-v) & v \end{bmatrix} \begin{bmatrix} c(u,0) \\ c(u,1) \end{bmatrix} - \begin{bmatrix} (1-u) & u \end{bmatrix} \begin{bmatrix} c(0,0) & c(0,1) \\ c(1,0) & c(1,1) \end{bmatrix} \begin{bmatrix} (1-v) \\ v \end{bmatrix} \tag{A.12}$$

The four boundary curves in equation (A.12) can be computed by the reconstruction method given in Appendix B. Those curves have C^1 continuities in their 2D projections as well as in the 3D reconstructions. Therefore, a patch constructed from such boundary curves will also have C^1 continuity across its boundaries.

A sheet of composite bicubic patches consists of a set of contiguous patches with at least C^1 continuity across their boundaries. The patches in a sheet are parametrized by a single parameter coordinate system P(u,v,w). This coordinate system [Figure A.1(b)] contains two sets of orthogonal lines which define an integer square grid. These lines, mapped on 3D surfaces, become the patch boundary curves and their intersections become patch control points. The lines define an absolute parameter coordinate system for $0 \leq \underline{u} \leq \underline{M}$, and $0 \leq \underline{v} \leq \underline{N}$, while within a single patch there is a relative parameter coordinate system for $0 \leq \underline{u}, \underline{v} \leq 1$. Using the notation of the absolute parameter coordinate system, a patch $\overline{g}_{m,n}(u,v)$ is defined by the four control points $h(m,n)$, $h(m+1,n)$, $h(m,n+1)$, and $h(m+1,n+1)$, or by the four boundary curves $c(m,v)$, $c(m+1,v)$, $c(u,n)$, and $c(u,n+1)$ for $\underline{m} \leq \underline{u} \leq \underline{m}+1$, and $\underline{n} \leq \underline{v} \leq \underline{n}+1$.

APPENDIX B

RECONSTRUCTION OF A PARAMETRIC CURVE FROM MULTIPLE PROJECTIONS

In Section 2.3 we reconstructed 3D surface points from multiple 2D perspective projections. These points were then used as the control points for parametric patches. In this appendix we develop an analogous method which reconstructs a 3D parametric space curve from its multiple 2D projections [Figure B.1]. Such a curve can directly become one of the four boundary curves of a surface patch.

A 3D cubic curve in an object coordinate system $O(x,y,z,1)_k$ is specified by:

(B.1)

$$\bar{c}(w)_k = \begin{bmatrix} x(w) \\ y(w) \\ z(w) \end{bmatrix}_k = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{bmatrix}_k \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix}$$

or, for short,

(B.2)

$$\bar{c}(w)_k = \bar{C}_k \, \bar{W}$$

Similarly, a 2D projection of this curve into an image coordinate system $Q(r,s,1)_i$ is:

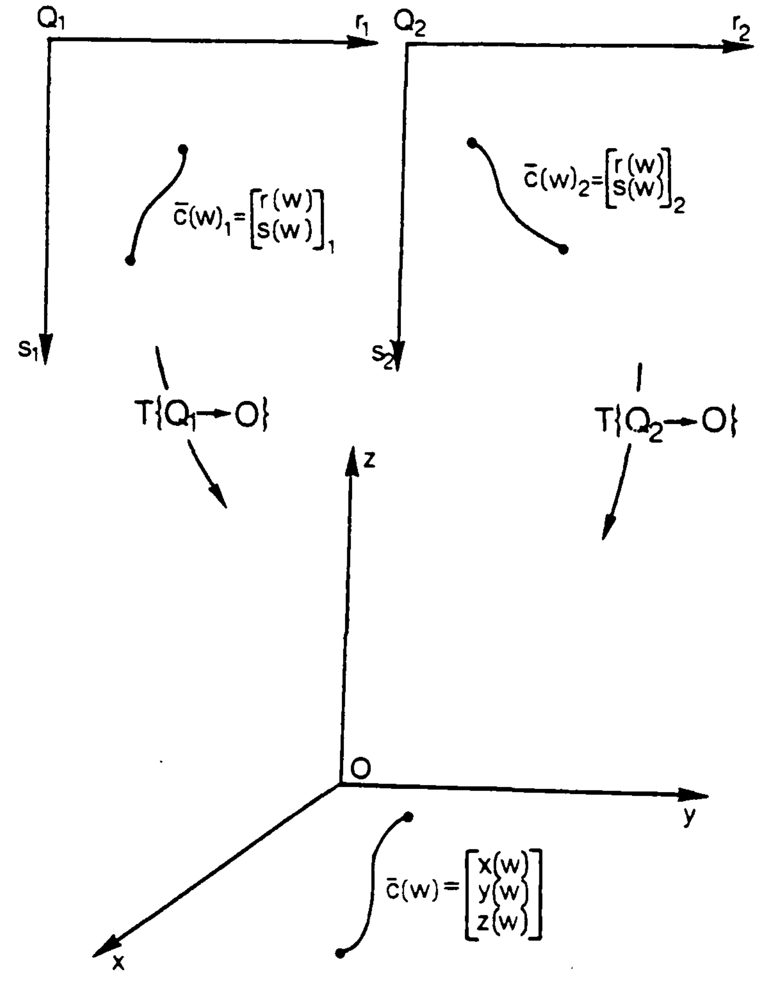

Q_1
r_1
s_1
$\bar{c}(w)_1 = \begin{bmatrix} r(w) \\ s(w) \end{bmatrix}_1$
Q_2
r_2
s_2
$\bar{c}(w)_2 = \begin{bmatrix} r(w) \\ s(w) \end{bmatrix}_2$
$T\{Q_1 \rightarrow O\}$
$T\{Q_2 \rightarrow O\}$
z
O
y
x
$\bar{c}(w) = \begin{bmatrix} x(w) \\ y(w) \\ z(w) \end{bmatrix}$

Figure B.1 Reconstruction of parametric curve $\bar{c}(w)$ in $\underline{O}(x,y,z,w)$ from its projections $\bar{c}(w)_1$ and $\bar{c}(w)_2$ in $Q(r,s,w)_1$ and $Q(r,s,w)_2$, respectively

$$\bar{c}(w)_i = \begin{bmatrix} r(w) \\ s(w) \end{bmatrix}_i = \begin{bmatrix} c_{11} & c_{12} & c_{23} & c_{24} \\ c_{21} & c_{22} & c_{23} & c_{24} \end{bmatrix}_i \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix} \tag{B.3}$$

or, for short,

$$\bar{c}(w)_i = \bar{C}_i \bar{W} \tag{B.4}$$

The mapping from the object coordinate system O_k to the image coordinate system Q_i is $T\{O_k \rightarrow Q_i\}$. Each perspective projection $\underline{i}$ of the 2D curve has two components: $r(w)_i$, and $s(w)_i$; the 3D curve has three components: $x(w)_k$, $y(w)_k$, and $z(w)_k$. A component of a cubic curve is constructed from four items of curve information. These are, typically, the position coordinates of the two end points $(h(0),h(1))$ and the slope of the curve at these points $(h_w(0),h_w(1))$. Here, $h(w)$ refers to the positional coordinate of the curve at $\underline{w}$, and $h_w(w)$ refers to the slope of the curve at $\underline{w}$. The parameter $\underline{w}$ is defined for $0 \leq \underline{w} \leq 1$.

A <u>Ferguson curve</u>, similar to a Ferguson patch of Appendix A, is constructed from the curve information, and a matrix of coefficients of blending functions:

(B.5)

$$c(w) = [\ h(0)\ \ h(1)\ \ h_w(0)\ \ h_w(1)\ ] \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix}$$

It is necessary to point out that the definition of the parameter $\underline{w}$ must be consistent in all projections $\underline{i}$; that is, a value of $\underline{w}$ yields 2D projections of the same point in 3D. The system of equations (2.3) can be now rewritten by the substitution of $x(w)_k$, $y(w)_k$, $z(w)_k$, $r(w)_i$, and $s(w)_i$ for x_k, y_k, z_k, r_i, and s_i, respectively, as:

(B.6)

$$\begin{bmatrix} d_{11}(w) & d_{12}(w) & d_{13}(w) & d_{14}(w) \\ d_{21}(w) & d_{22}(w) & d_{24}(w) & d_{24}(w) \end{bmatrix} \begin{bmatrix} x(w) \\ y(w) \\ z(w) \\ 1 \end{bmatrix} = \overline{0}$$

where $d_{1p}(w)$ and $d_{2p}(w)$ are defined by:

(B.7)

$$d_{1p}(w) = t_{3p}\ (\ c_{11i}w^3 + c_{12i}w^2 + c_{13i}w + c_{14i}\ ) - t_{1p}$$
$$d_{2p}(w) = t_{3p}\ (\ c_{21i}w^3 + c_{22i}w^2 + c_{23i}w + c_{24i}\ ) - t_{2p}$$

for $\underline{p}$ = 1,2,3,4.

Also, all $\underline{c}$'s and $\underline{t}$'s in equation (B.7) apply to transformation $\underline{i}$, and $x(w)$, $y(w)$, and $z(w)$ are defined in equation (B.1). We need to solve the system of equations in (B.6) for the twelve coefficients of $\overline{C}_k$ given in (B.1). The matrix elements of equation (B.7) and the vector elements of equation (B.1) are multiplied in equation (B.6) and the coefficients of the curve (B.1) are factored out to give the following system of two linear equations with twelve unknowns:

$$\begin{bmatrix} d_{11}(w)w^3 & d_{11}(w)w^2 & \ldots & d_{13}(w)w & d_{13}(w) \\ d_{21}(w)w^3 & d_{22}(w)w^2 & \ldots & d_{23}(w)w & d_{23}(w) \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{12} \\ \ldots \\ \ldots \\ \ldots \\ \ldots \\ c_{33} \\ c_{34} \end{bmatrix} = \begin{bmatrix} d_{14}(w) \\ d_{24}(w) \end{bmatrix} \tag{B.8}$$

Each of the two equations in (B.8) can be expanded into four linearly independent equations by substituting four linearly independent vectors $\overline{W}$ (values $\underline{w}_1$, $\underline{w}_2$, $\underline{w}_3$, and $\underline{w}_4$) for the parameter $\underline{w}$, because the cubic basis vector $\overline{W}$ yields four linearly independent vectors. Therefore, for a single projection $\underline{i}$ there is a system of eight equations with twelve unknowns, expressed as:

(B.9)

$$\begin{bmatrix} d_{11}(w_1)w_1^3 & \cdots & d_{13}(w_1) \\ d_{11}(w_2)w_2^3 & \cdots & d_{13}(w_2) \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ d_{21}(w_3)w_3^3 & \cdots & d_{23}(w_3) \\ d_{21}(w_4)w_4^3 & \cdots & d_{23}(w_4) \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{12} \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ c_{33} \\ c_{34} \end{bmatrix} = \begin{bmatrix} d_{14}(w_1) \\ d_{14}(w_2) \\ \cdots \\ \cdots \\ d_{24}(w_3) \\ d_{24}(w_4) \end{bmatrix}$$

or, for short,

(B.10)

$$\overline{D}\ \overline{C}' = \overline{D}'$$

In general, for $I \geq 2$ projections, there is a system of $8 \cdot I$ equations of twelve unknowns:

(B.11)

$$\begin{bmatrix} \overline{D}_1 \\ \overline{D}_i \\ \overline{D}_I \end{bmatrix} \overline{C} = \begin{bmatrix} \overline{D}'_1 \\ \overline{D}'_i \\ \overline{D}'_I \end{bmatrix}$$

System (B.11) is solved by the least-squares method which yields:

(B.12)

$$\begin{bmatrix} \overline{D}_1 \\ \overline{D}_i \\ \overline{D}_I \end{bmatrix}^t \begin{bmatrix} \overline{D}_1 \\ \overline{D}_i \\ \overline{D}_I \end{bmatrix} \overline{C}' = \begin{bmatrix} \overline{D}_1 \\ \overline{D}_i \\ \overline{D}_I \end{bmatrix}^t \begin{bmatrix} \overline{D}'_1 \\ \overline{D}'_i \\ \overline{D}'_I \end{bmatrix}$$

This curve reconstruction method is particularly useful when the 2D projections of the curve are computed from information other then four position points, i.e., two end points and two slopes as in equation (B.5), or when the points in one projection cannot be matched with their corresponding points in the other projections. This method can be generalized to parametric curves with an arbitrary-order basis vector $\overline{W}$.

APPENDIX C

SURFACE REPRESENTATIONS AND OPERATIONS

This appendix contains a summary of representations of the four types of 3D surface elements (planar, spherical, quadric, and bicubic) and the five basic geometric operations (3D rigid transformation, surface-normal vector, surface-normal curvature, intersection with 3D line, and surface existence) on them. This is followed by individual descriptions of the representations and the operations for each type of surface elements.

C.1 Summary

C.1.1 Representations of Surface Elements

(C.1)

(a) algebraic: $f(x,y,z,w) = 0$

bounds: other algebraic representations arranged in a boolean tree

(C.2)

(b) parametric: $\overline{g}(u,v) = \left[x(u,v),\ y(u,v),\ z(u,v),\ w(u,v) \right]$

bounds: limits on parameters $\underline{u}$ and $\underline{v}$

C.1.2 Geometric Operations

(a) transformation from object coordinate space $O(x,y,z,w)_k$ to object coordinate space $O(x,y,z,w)_{k'}$ by $T\{O_k \rightarrow O_{k'}\}$:

(C.3)

$$T\{O_k \rightarrow O_{k'}\} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation $T\{O_k \rightarrow O_{k'}\}$ is a function of the following transformation parameters:

(C.4)

translations:	tx, ty, tz
rotations:	rx, ry, rz
translations of rotations:	trx, try, trz
order of rotations:	irx, iry, irz
scale factors:	sx, sy, sz
translations of scale factors:	tsx, tsy, tsz

The transformation parameters are normally written in matrix form. The translation matrix is:

(C.5)

$$\bar{T}(tx,ty,tz) = \begin{bmatrix} 1 & 0 & 0 & -tx \\ 0 & 1 & 0 & -ty \\ 0 & 0 & 1 & -tz \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

the rotation matrices, in the irz, iry, and irx order of precedence are:

$$\overline{R}(rx,ry,rz) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(rx) & \sin(rx) & 0 \\ 0 & -\sin(rx) & \cos(rx) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(ry) & 0 & -\sin(ry) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(ry) & 0 & \cos(ry) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(rz) & \sin(rz) & 0 & 0 \\ -\sin(rz) & \cos(rz) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{C.6}$$

and the scale matrix is:

$$\overline{S}(sx,sy,sz) = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{C.7}$$

The complete transformation is then a concatination of scale transformations around point $\overline{h}(tsx,tsy,tsz)$, followed by z,y,x rotations around point $\overline{h}(trx,try,trz)$, followed by translation to point h(tx,ty,tz):

$$T\{O_k \rightarrow O_{k'}\} = \overline{T}(tx,ty,tz) \overline{T}(-trx,-try,-trz)\overline{R}(rx,ry,rz)\overline{T}(trx,try,trz) \tag{C.8}$$

$$\overline{T}(-tsx,-tsy,-tsz)\overline{S}(sx,sy,sz)\overline{T}(tsx,tsy,tsz)$$

The inverse matrix $T\{O_k \rightarrow O_{k'}\}^{-1}$ transforms from object coordinate space $O(x,y,z,w)_{k'}$ to object coordinate space $O(x,y,z,w)_k$, that is:

(C.9)

$$T\{O_k \rightarrow O_{k'}\}^{-1} = T\{O_{k'} \rightarrow O_k\}$$

(b) surface-normal vector at point $\overline{h}(x,y,z)$ of algebraic representation $f(x,y,z)$:

(C.10)

$$\overline{N} = \left[df/dx \;\; df/dy \;\; df/dz \right]$$

surface-normal vector at point $\overline{h}(u,v)$ of parametric representation $\overline{g}(u,v)$:

(C.11)

$$\begin{aligned}\overline{N} &= \overline{g}(u,v)_u \times \overline{g}(u,v)_v \\ &= \left[dx/du \;\; dy/du \;\; dz/du \right] \times \left[dx/dv \;\; dy/dv \;\; dz/dv \right]\end{aligned}$$

(c) <u>normal</u> surface curvature <u>q</u> at point $\overline{h}(u,v)$ of parametric representation $\overline{g}(u,v)$ in the direction of the surface curve $\overline{g}(u(t),v(t))$ is given by:

(C.12)

$$q = \frac{D_1\dot{u}^2 + 2D_2\dot{u}\dot{v} + D_3\dot{v}^2}{E_1\dot{u}^2 + 2E_2\dot{u}\dot{v} + E_3\dot{v}^2}$$

where

$$\dot{u} = \frac{du(t)}{dt}, \text{ and } \dot{v} = \frac{dv(t)}{dt}$$

$$D_1 = \overline{N} \cdot \overline{g}(u,v)_{uu}$$

$$D_2 = \overline{N} \cdot \overline{g}(u,v)_{uv}$$

$$D_3 = \overline{N} \cdot \overline{g}(u,v)_{vv}$$

$$E_1 = \overline{g}(u,v)_u \cdot \overline{g}(u,v)_u$$

$$E_2 = \overline{g}(u,v)_u \cdot \overline{g}(u,v)_v$$

$$E_3 = \overline{g}(u,v)_v \cdot \overline{g}(u,v)_v$$

(d) intersection with a 3D parametric line represented by:

(C.13)

$$\overline{l}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ w(t) \end{bmatrix} = \begin{bmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \\ l_{31} & l_{32} \\ l_{41} & l_{42} \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix}$$

or, for short,

(C.14)

$$\overline{l}(t) = \overline{L}\,\overline{t}$$

The transformation of line $\overline{L}$ by $T\{O_k \rightarrow O_{k'}\}$ is:

(C.15)

$$\overline{L}' = T\{O_k \rightarrow O_{k'}\}\,\overline{L}$$

C.2 Planar Face

(C.16)

algebraic: $$f(x,y,z) = a_0x + a_1y + a_2z + a_3 = 0$$

$$= \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

bounds: lists of edges and vertices [Figure C.1(a)]

(C.17)

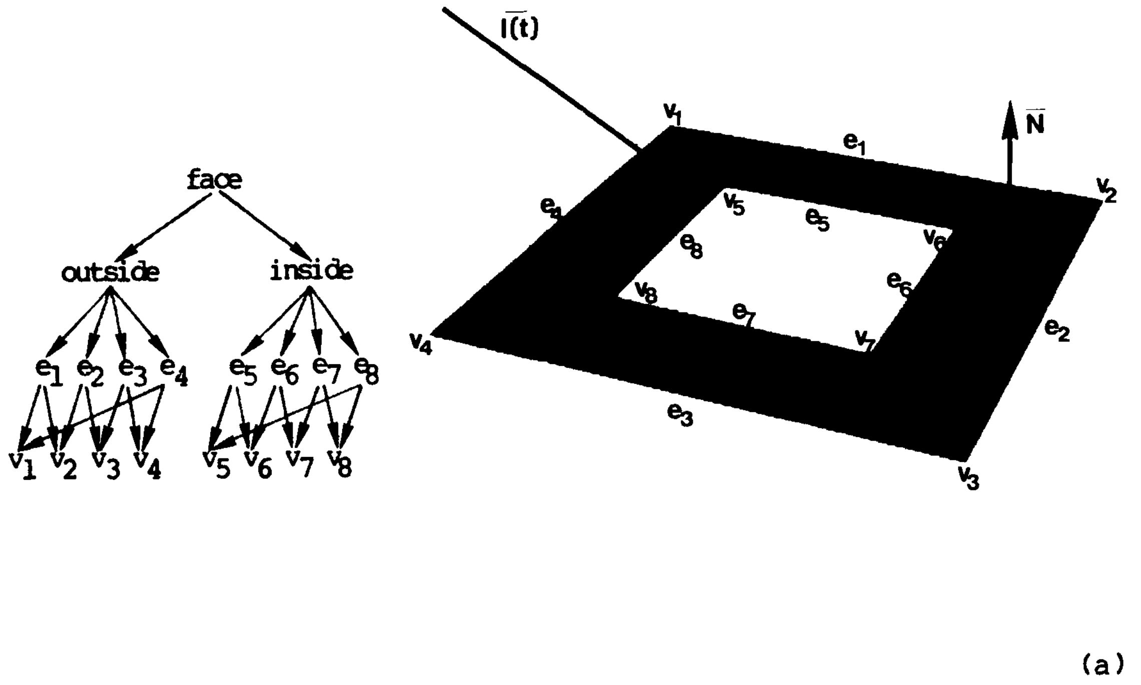
parametric: $$g(u,v) = \begin{bmatrix} b_0 & b_1 & b_2 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

bounds: limits on parameters $\underline{u}$ and $\underline{v}$, possibly combined with lists of edges and vertices

(C.18)

transformation: $$\begin{bmatrix} a_0' \\ a_1' \\ a_2' \\ a_3' \end{bmatrix} = T\{O \rightarrow O'\} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

(the algebraic representation in equation (C.16) is pre-multiplied by $T\{O \rightarrow O'\}$)



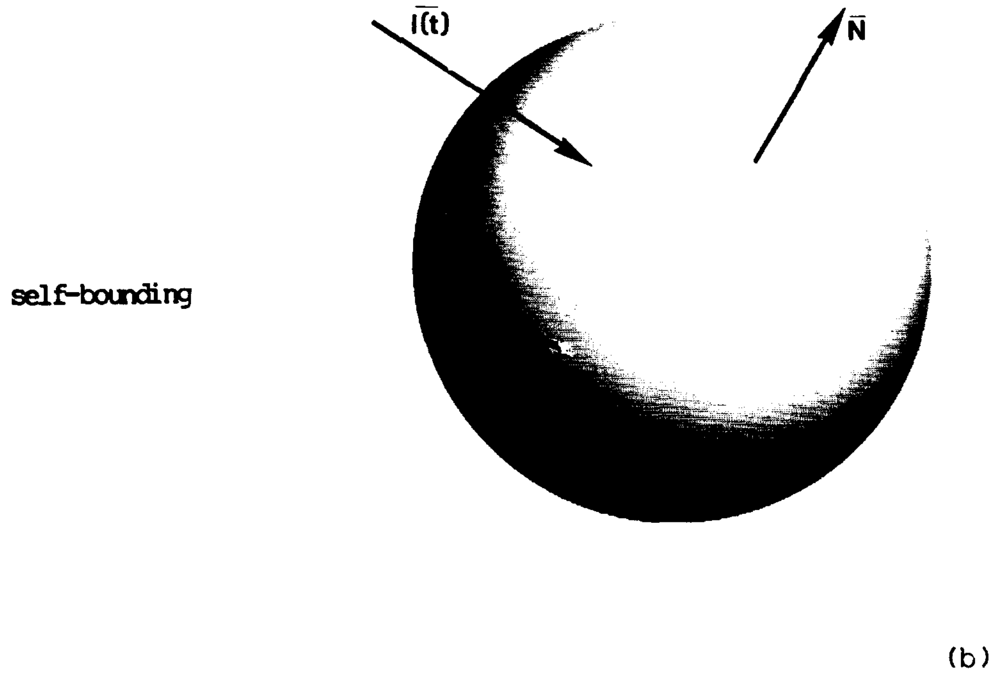


Figure C.1 Surface representation: (a) planar face, (b) sphere

(C.19)

normal vector: $\bar{N} = [a_0\ a_1\ a_2]$

or

$\bar{N} = [b_{0x}\ b_{0y}\ b_{0z}] \times [b_{1x}\ b_{1y}\ b_{1z}]$

(C.20)

intersection: $f(l_x(t), l_y(t), l_z(t)) = 0$

(obtained by substitution of parametric line representation (C.13) into algebraic surface representation (C.16), a linear equation with up to one possible solution [Figure C.1(a)]; the number of valid intersections of a line from the point of intersection to infinity in the plane with all edges of the face then determines whether the intersection point is inside or outside the planar face: if the number is odd the point is inside, otherwise it is outside; this method allows a planar face bounded by a closed list of edges to contain any number of not-intersecting holes or protrusions, each similarly bounded by a closed list of edges)

C.3 Sphere

(C.21)

algebraic: $f(x,y,z) = (x-x_c)^2+(y-y_c)^2+(z-z_c)^2-r^2 = 0$

bounds: none - the representation is self-bounding

(C.22)

parametric: $x(u,v) = x_c + r\cos(u)\cos(v)$

$y(u,v) = y_c + r\cos(u)\sin(v)$

$z(u,v) = z_c + r\sin(u)$

bounds: $-\underline{pi}/2 \leq \underline{u} \leq \underline{pi}/2$

$0 \leq \underline{v} \leq 2\ \underline{pi}$

(C.23)

transformation: $$\begin{bmatrix} x_c' \\ y_c' \\ z_c' \\ w_c' \end{bmatrix} = T\{O \rightarrow O'\} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

(the center point $\bar{h}(x_c,y_c,z_c,w_c)$ of the sphere is pre-multiplied by $T\{O \rightarrow O'\}$)

(C.24)

normal vector: $\bar{N} = \begin{bmatrix} x-x_c & y-y_c & z-z_c \end{bmatrix}$

or

$$\bar{N} = \begin{bmatrix} -\sin(u)\ \cos(v) \\ -\sin(u)\ \sin(v) \\ \cos(u) \end{bmatrix} \times \begin{bmatrix} -\cos(u)\ \sin(v) \\ \cos(u)\ \cos(v) \\ 0 \end{bmatrix}$$

(C.25)

intersection: $f(l_x(t), l_y(t), l_z(t)) = 0$

(obtained by substitution of parametric line representation (C.13) into algebraic surface representation (C.21), a quadric equation with up to two possible solutions [Figure C.1(b)]; if the sphere serves as a bounding volume it is not necessary to compute the exact intersection points, therefore only the line to center-of-sphere distance is computed and compared with the radius of the sphere)

C.4 Quadric Surface

(C.26)

algebraic:

$$f(x,y,z) = a_0x^2 + a_1y^2 + a_2z^2 + a_3xy + a_4xz + a_5yz + a_6x + a_7y + a_8z + a_9 = 0$$

$$= \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} a_0 & a_3/2 & a_4/2 & a_6 \\ a_3/2 & a_1 & a_5/2 & a_7 \\ a_4/2 & a_5/2 & a_2 & a_8 \\ a_6 & a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x & y & z & 1 \end{bmatrix} \bar{A} \begin{bmatrix} x & y & z & 1 \end{bmatrix}^t$$

bounds: boolean tree of quadric surfaces [41]

(C.27)

parametric:

$$g(u,v) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & b_{13} \\ 0 & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} v^2 \\ v \\ 1 \end{bmatrix}$$

$$= \bar{U}\,\bar{B}\,\bar{V}^t$$

bounds: limits on parameters <u>u</u> and <u>v</u>

(C.28)

transformation: $\bar{A}' = T\{O \rightarrow O'\}^{-1} \; \bar{A} \; (T\{O \rightarrow O'\}^{-1})^t$

(matrix $\overline{A}$ of equation (C.26) is pre-multiplied by the inverse of T{O→O'} and post-multiplied by the transpose of the inverse of T{O→O'} [41])

(C.29)

normal vector:

$$\overline{N} = \begin{bmatrix} 2a_0x + a_3y + a_4z + a_6 \\ 2a_1y + a_3x + a_5z + a_7 \\ 2a_2z + a_4x + a_5y + a_8 \end{bmatrix}$$

or

$$\overline{N} = \begin{bmatrix} (2b_{13}u+b_{22}v+b_{23})_x \\ (2b_{13}u+b_{22}v+b_{23})_y \\ (2b_{13}u+b_{22}v+b_{23})_z \end{bmatrix} \times \begin{bmatrix} (2b_{31}v+b_{22}u+b_{32})_x \\ (2b_{31}v+b_{22}u+b_{32})_y \\ (2b_{31}v+b_{22}u+b_{32})_z \end{bmatrix}$$

(C.30)

intersection: $f(l_x(t), l_y(t), l_z(t)) = 0$

(obtained by substitution of parametric line representation (C.13) into algebraic surface representation (C.26), a quadric equation with up to two possible solutions [Figure C.2(a)])

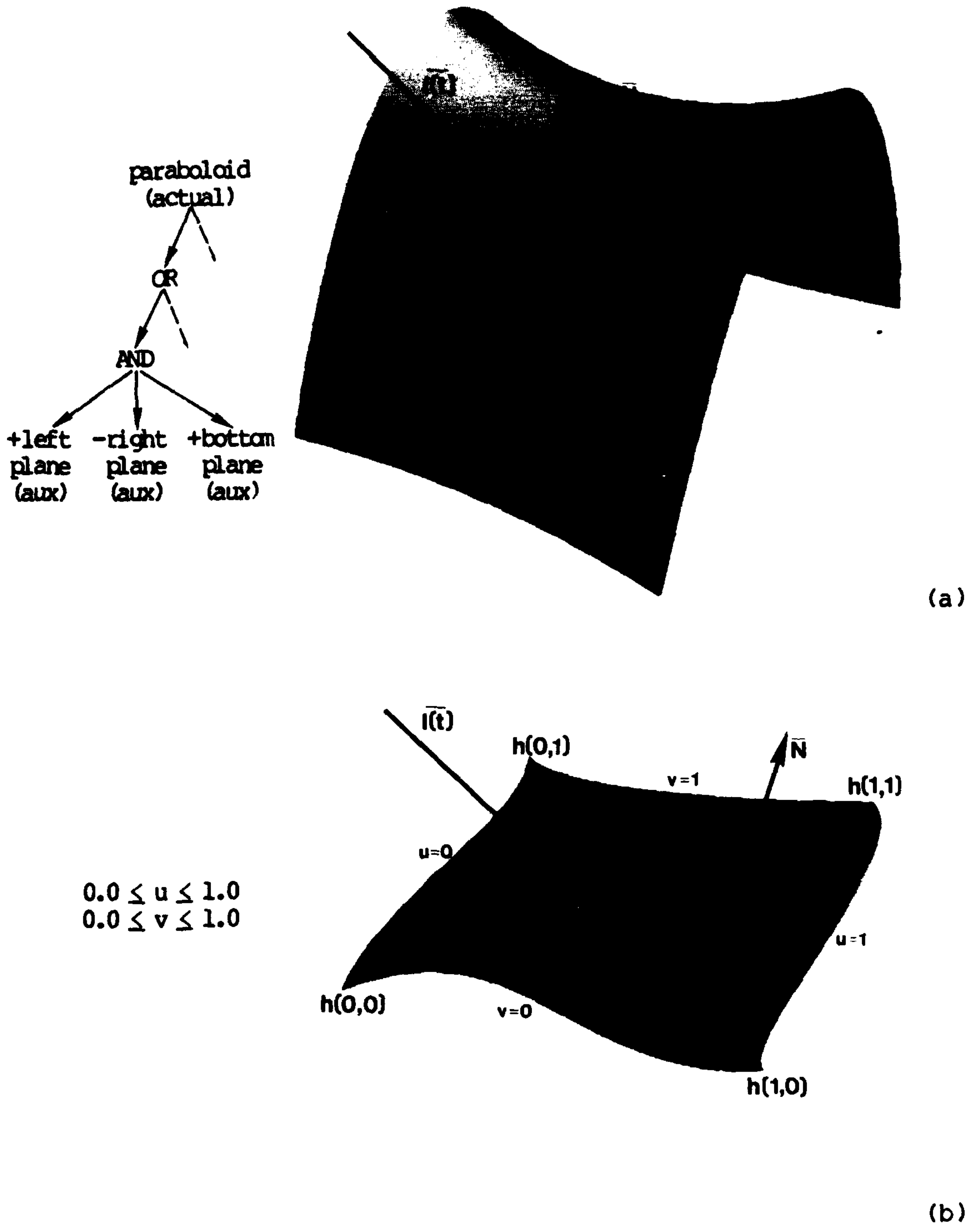


Figure C.2 Surface representation: (a) quadric surface, (b) bicubic patch

C.5 Bicubic Patch

algebraic: none

(C.31)

parametric:
$$g(u,v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

$$= \overline{U}\,\overline{B}\,\overline{V}^t$$

bounds: $0 \leq \underline{u}, \underline{v} \leq 1$ in relative parameter coordinates

(C.32)

transformation:
$$g_x(u,v)' = T\{O \rightarrow O'\}\ g_x(u,v)$$
$$g_y(u,v)' = T\{O \rightarrow O'\}\ g_y(u,v)$$
$$g_z(u,v)' = T\{O \rightarrow O'\}\ g_z(u,v)$$

(matrix $\overline{B}$ of each component in equation (C.31) is pre-multiplied by $T\{O \rightarrow O'\}$)

(C.33)

normal vector: $\overline{N} = \overline{g}(u,v)_u \times \overline{g}(u,v)_v$

(C.34)

intersection: $\overline{g}(u,v) - \overline{I}(t) = 0$

or when written as three simultaneous cubic equations:

(C.35)

$$g_x(u,v) - l_x(t) = 0$$
$$g_y(u,v) - l_y(t) = 0$$
$$g_z(u,v) - l_z(t) = 0$$

This system of non-linear equations is numerically solved by a modified Newton-Raphson method with tests for equation convergence to the nearest intersection solution, and constraints. limiting the parameters to within the patch and the bounding volume [Figure C.2(b)]. Intersection of the line with a planar approximation of the patch or its subpatch provides the initial estimate of the three unknowns.

The solution proceeds as follows; let:

(C.36)

$$E(t,u,v) = g_x(u,v) - l_x(t)$$
$$F(t,u,v) = g_y(u,v) - l_y(t)$$
$$G(t,u,v) = g_z(u,v) - l_z(t)$$

Then by first-order Taylor expansion of the system of equations in (C.36) we can incrementally solve for the parameters t, u, and v:

(C.37)

$$t_{n+1} = t_n - dt = t_n - \frac{\begin{vmatrix} E & dE/du & dE/dv \\ F & dF/du & dF/dv \\ G & dG/du & dG/dv \end{vmatrix}}{|\, D(t,u,v) \,|}$$

(C.38)

$$u_{n+1} = u_n - du = u_n - \frac{\begin{vmatrix} dE/dt & E & dE/dv \\ dF/dt & F & dF/dv \\ dG/dt & G & dG/dv \end{vmatrix}}{|\, D(t,u,v) \,|}$$

(C.39)

$$v_{n+1} = v_n - dv = v_n - \frac{\begin{vmatrix} dE/dt & dE/du & E \\ dF/dt & dF/du & F \\ dG/dt & dG/du & G \end{vmatrix}}{|\, D(t,u,v) \,|}$$

where $|D(t,u,v)|$ is the determinant of the Jacobian matrix defined by:

(C.40)

$$D(t,u,v) = \begin{bmatrix} dE/dt & dE/du & dE/dv \\ dF/dt & dF/du & dF/dv \\ dG/dt & dG/du & dG/dv \end{bmatrix}$$

An outline of the algorithm follows:

(C.41)

```
procedure csystem(t,u,v);
   begin
   [test magnitudes  of the three equations]
```

```
while |E(t,u,v)| > $eps_1$ |
      |F(t,u,v)| > $eps_1$ |
      |G(t,u,v)| > $eps_1$ do
   begin
   [test for vanishing determinant of
   Jacobian matrix]
   dd := determinant(D(t,u,v));
   if |dd| < $eps_2$ then return(fail);
   dt := [from equation (C.37)];
   du := [from equation (C.38)];
   dv := [from equation (C.39)];
   [test boundary condition, if outside
   clip to boundary values, if previously
   on boundary then fail]
   if (t-dt > $t_{max}$) | (t-dt < $t_{min}$) |
      (u-du > $u_{max}$) | (u-du < $u_{min}$) |
      (v-dv > $v_{max}$) | (v-dv < $v_{min}$) then
      begin
      if (t = $t_{min}$) | (t = $t_{max}$) |
         (u = $u_{min}$) | (u = $u_{max}$) |
         (v = $v_{min}$) | (v = $v_{max}$) then
                                  return(fail);
         else clip to boundary values;
      [test convergence to the nearest
      solution, if increments become too
      small exit with failure]
      while
         |E(t-dt,u-du,v-dv)|>|E(t,u,v)| |
         |F(t-dt,u-du,v-dv)|>|F(t,u,v)| |
         |G(t-dt,u-du,v-dv)|>|G(t,u,v)|
         do
         begin
         dt := dt / 2;
         du := du / 2;
         dv := dv / 2;
         [test if at local minimum or
         maximum]
         if $dt^2+du^2+dv^2 \leq eps_3$
                   then return(fail);
         end;
      end;
   [compute new increment of the solu-
   tion]
   t := t - dt;
   u := u - du;
   v := v - dv;
   end;
return(t,u,v);
end;
```

APPENDIX D

SOFTWARE COMMANDS

This appendix contains descriptions of the available commands in the four software command processors, SURE, SUED, ALIGN, and STRAW, described in Chapter 5. The syntax of the commands is given by:

(D.1)

```
COMMAND: <parameter_list>;
```

where

```
<parameter_list> ::= <parameter_list> <operator>
                     <parameter> | <parameter>
<parameter>      ::= <identifier> | <constant> |
                     {keyword}
<operator>       ::= ' ' | ',' | ' ' | '|' | '+' | '-'
                     | ';'
```

Here, <identifier> refers to the name of data in a processor's data base, or to the name of a directory or a file; <constant> refers to a numerical constant in either integer or floating-point format; and {keyword} contains a command's keyword.

D.1 Processor SURE

```
TAPEIN: <directory_name>, <8-bit_file_name>,
```

<16-bit_file_name>;

reads a digitized image from the current position of a magnetic tape and creates an 8 bit/pixel and, optionally, a 16 bit/pixel image file.

HISTOGRAM: <directory_name>, <file_name>;

computes a histogram of a 16 bit/pixel, 8 bit/pixel or 1 bit/pixel (binary) image.

CONVERT: <directory_name>, <in_file_name>, <out_file_name>;

converts either a 16 bit/pixel image to an 8 bit/pixel image, or an 8 bit/pixel image to a binary image.

WINDOW: <directory_name>, <in_file_name>, <out_file_name>;

creates a window image from an input image.

FFT2D: <directory_name>, <in_file_name>, <out_file_name>;

computes the 2D fast Fourier transform of an 8 or 16 bit/pixel image.

FILTER: <filter_parameters>;

filters an image in the frequency domain.

IFFT2D: <directory_name>, <in_file_name>, <out_file_name>;

computes the 2D inverse fast Fourier transform and generates an 8 or 16 bit/pixel image.

MFFT2D: <directory_name>, <in_file_name>, <out_file_name>;

converts a 2D Fourier transform of an image into a 8 bit/pixel image of logarithmically scaled magnitudes of the Fourier transform.

CLEAN: <directory_name>, <in_file_name>, <out_file_name>;

smooths either an 8 bit/pixel or a binary image with a 3 x 3 or a 5 x 5 pixel operator in the spatial domain.

PLOT: <directory_name>, <in_file_name>;

plots an 8 bit/pixel or a binary image on the electrostatic printer-plotter. The 8 bit/pixel image is dithered.

COPY: <directory_name>, <in_file_name>;

copies a window of an 8 bit/pixel or a binary image to the frame buffer of the raster terminal.

LOAD: <identifier>, <directory_name>, <file_name>;

loads an image or partially processed data into SURE.

EXTRACT: <identifier>;

extracts the image projections of the corners of camera calibration marks from a binary image.

MATCH_MARKS: <identifier>;

matches the 2D image projections of the corners of camera calibration marks with their 3D coordinates.

TRANSFORMATION: <identifier>;

computes the image transformation matrix by matching a 3D model of the calibration marks with their 2D projections in the specified image.

THIN: <identifier>;

iteratively thins a binary image until a line pixel has at most only two neighbors.

RECONSTRUCT: <image_1>, <image_2>, ... ;

reconstructs a 3D surface representation from information obtained from two or more images.

SAVE: <identifier>, <directory_name>, <file_name>;

saves an image or partially processed data from SURE to a file.

LIST: {marks|grid|nodes|curves};

lists contents of the specified data.

DISPLAY: <identifier>;

displays the specified data on the vector-drawing graphics terminal.

D.2 Processor SUED

SPHERE: <identifier>, <color>, <x>, <y>, <z>, <radius>;

defines the name, color, location and size of a sphere.

VERTEX: <identifier>, <x>, <y>, <z>;

defines a 3D point that can be used either as a vertex point in planar faces or as a control point in bicubic patches.

PLANE: <identifier>, <color>, <vertex_list> & <vertex_list> & .... ;

defines a planar face that is bounded by a list of vertices specifing outer edges, and, optionally, one or more lists of vertices specifing inner edges (holes). A bounding volume is cast around the planar face.

POLYHEDRON: <identifier>, <plane>, <plane> .... ;

defines a set of planar faces which form a polyhedron-like object (holes in faces are allowed and the object does not have to be closed). A bounding volume is cast around the object.

PATCH: <identifier>, <color>, <list_of_control_points>, <r_min>;

defines a bicubic patch that is fitted into 16 3D control points which are defined by the VERTEX command. Such a patch will usually have at most C^0 continuity with any adjacent patches.

QUADRIC_COEFFS: <identifier>, <color>, {actual|auxiliary}, $<a_0>$, $<a_1>$, ... $<a_9>$;

defines the name, color, type and algebraic representation of a quadric surface. The type of the surface may be auxiliary - used only as a bounding surface - or normal.

QUADRIC_BOUNDS: <identifier>, <bounding_tree>;

defines a single level AND-OR logical tree of bounding quadric surfaces.

COLOR: <identifier>, <r_ambient>, <g_ambient>, <b_ambient>, <ambient_coeff>, <diffuse_coeff>, <reflection_coeff>, <transmission_coeff>, <refraction_coeff>, <specular_coeff>,

<glossiness_coeff>;

defines the ambient red, green and blue intensities, and the ambient, diffuse, reflection, transmission, refraction, specular and glossiness coefficients of a color.

LOAD: <identifier>, <directory_name>, <file_name>;

loads a surface representation from a file into the data base of the processor.

SAVE: <identifier>, <directory_name>, <file_name>;

stores a surface representation into a file from the data base of the processor.

SYMMETRY: <identifier>, <from_x>, <from_y>, <from_z>, <to_x>, <to_y>, <to_z>, <p_increment>, <c_increment>;

attempts to convert a sheet of bicubic patches into a generalized cylinder-like shape.

QUADTREE: <identifier_quadtree>, <identifier_patches>;

converts a sequential list of bicubic patches into a structured quadtree of the patches with bounding volumes and normal vectors.

SET_DISPLAY: {initialize}, {blink|noblink}, {fast|normal}, {solid|dotted|dashed}, {camera_stand|maximum};

defines the display mode of the subsequently displayed data.

DISPLAY: {camera_stand|nodes|patches|quadtrees|cylinders};

displays the specified part of the currently defined data.

STATUS_DISPLAY;

lists the current status of the display device.

LIST: {nodes|patches|quadtrees|cylinders};

lists the specified part of the currently defined data.

VERIFY: {nodes|patches|quadtrees|cylinders};

verifies the integrity of the specified data.

MODIFY: <node>, <x>, <y>, <z>, <pointer_list>;

adds a new node or modifies the contents of an existing node of a parametric sheet.

SMOOTH: <sheet>, <u_order>, <v_order>;

smooths a sheet of bicubic patches with an orthogonal bivariate polynomial of the specified orders.

TRANSFORM: <identifier>, <tx>, <ty>, <tz>, <rx>, <ry>, ...;

creates a 3D transformation matrix composed from translations, rotations in arbitrary order around an arbitrary point, and scale changes around an arbitrary point.

RELATION: <identifier>, <parameters>;

creates a relational-node entry.

BOUND: <identifier>, <parameters>;

creates a boundary-volume entry.

INSERT: <identifier>, <node>, <transform>;

inserts a graph node into the graph data structure.

INITIALIZE; STATUS; HELP; EXIT;

are the obvious and self-explanatory utility commands.

D.3 Processor ALIGN

LOAD_SURFACE: <identifier>, <directory_name>, <file_name>;

loads the first or the second structured surface into the data base.

SET_TRANSFORM: {tx|no_tx}, <max_tx>, <min_tx>, ... ;

defines transformation parameters which can be modified during the search process and sets limits of their ranges.

MODE: {debug|nobug}, {display|noplay};

enables and disables debug and display modes of the search process.

MATCH;

aligns the two currently defined surfaces into a common object coordinate system.

MERGE;

merges the two aligned surfaces, defined in the data base, into a common parameter coordinate space.

SAVE_SURFACE: <directory_name>, <file_name>;

saves the aligned and merged surfaces into a file.

LOAD_SEARCH: <directory_name>, <file_name>;

loads the search data base generated by the ALIGN command from a file.

SAVE_SEARCH: <directory_name>, <file_name>;

saves the search data base to a file.

TRACE: <search_node>;

traverses the search tree of the alignment process to the specified node and displays the surface transformation of each node in the path.

TRACE_GOAL;

traverses the search tree of the alignment process to the goal node and displays the surface transformation of each node in the path.

REPLAY;

repeats a previously executed alignment process by traversing the nodes of its search tree in the order in which they were generated.

INITALIZE; STATUS; HELP; EXIT;

are again the obvious, indispensable, and self-explanatory utility commands.

D.4 Processor STRAW

CAMERA: <x>, <y>, <z>, <rot_x>, <rot_y>, <rot_z>, <focal_length>, {perspective|orthographic};

defines the position of the camera model (center of projection) in the object coordinate system, rotations of the optical axis of the camera model, the focal length of its lens, and the type of projection. In the orthographic projection the focal length only scales the rays' parameter; all rays are parallel to each other and perpendicular to the image plane.

FRAME: <r_size>, <s_size>, <r_pixels>, <s_lines>, <i_pixel>, <f_pixel>, <i_line>, <f_line>;

defines the size of the image frame in object coordinate system units, in pixel units, and defines window in the image to be actually computed. The size of the image plane and the focal length of the camera model together define the viewing pyramid.

SAMPLE: <levels>, <subdivision_coeffs>;

specifies a maximum number of levels that a pixel can be subdivided to perform image antialiasing. A subdivision coefficient for each level specifies the amount by which the average intensity of a pixel area can differ from any one of its samples before the area is subdivided.

SHADE: <max_ray>, <min_hide>, <min_shadow>, <max_reflect>, <max_transmit>;

defines the maximum number of nodes in the shading tree, the minimum distances that a valid ray intersection must be from the origin of a ray to prevent false ray bounces and false shadows caused by round-off errors, and linear intensity attenuation for reflected and refracted rays.

IMAGE_OUTPUT: <directory_name>, <file_name>;

creates a file where the generated raster image will be written by the EXECUTE command.

POINT_OUTPUT: <directory_name>, <file_name>;

creates a file where the image sample points will be written by the EXECUTE command. This file can be later converted by a post-processor into a raster image. Post-processor FOCUS generates images which are focused and have a depth of field; post-processor BLUR adds motion blur.

PAINT_TABLE: <identifier>, [<from_x>, <from_y>, <from_z>,
<from_red>, <from_green>, <from_blue>,
<to_x>, <to_y>, <to_z>,
<to_red>, <to_green>, <to_blue>];

defines a paint table which contains a sequence of 3D vectors specified by their initial and final points and corresponding intensity values. The intensity values are linearly interpolated along the direction of the vector and used as ambient intensity to be painted on a surface.

INTENSITY_MAP: <identifier>, <directory_name>, <file_name>;

defines a color image which is to be used as an ambient intensity map of a color.

TEXTURE_MAP: <identifier>, <directory_name>, <file_name>;

defines a monochrome image which is to be used as a texture function of a color.

COLOR: <identifier>, <color_coefficients> & <image_map>,
<image_map_coefficients> & <texture_map>,
<texture_map_coefficients> & <paint_table>;

defines the color properties of a surface element. This command is identical to the COLOR command in SUED but the source of ambient intensity can also be a color image or a paint table, and a texture function defined in a monochrome image may also be added.

LIGHT: <identifier>, <x>, <y>, <z>, <red>, <green>, <blue>,
{shadow|noshadow}, <contrast>;

defines the name, position, color and intensity of a point-light source which illuminates the scene. Optionally, shadows cast by the light source can be computed with a specified contrast.

LOAD: <directory_name>, <file_name>;

loads a structured surface file generated by the SUED processor.

PATCH_TREES: <identifier>, <color>, <directory_name>,
<file_name>;

loads a file of structured bicubic-patch quadtrees from a file.

SPHERE: <parameter_list>;

VERTEX: <parameter_list>;

PLANE: <parameter_list>;

POLYHEDRON: <parameter_list>;

PATCH: <parameter_list>;

QUADRIC_COEFFS: <parameter_list>;

QUADRIC_BOUNDS: <parameter_list>;

enter unstructured surface elements as in the SUED processor.

STATUS: <what>;

lists the current status and contents of the specified data in the processor.

LIST: <what>;

defines the parts of STRAW data that are to be listed during image generation.

INITIALIZE;

initializes the STRAW data base.

EXECUTE;

computes a synthetic image and writes it to the currently opened output image and sample files. In an animated sequence any part of the data base can be modified between successive EXECUTE commands.

EXIT;

terminates execution of the processor.

STEREO: {left|right}, <view_x>, <view_y>, <view_z>, <angle>;

replaces the current camera parameters by those of a complementary stereo view computed from the specified view point and angle of separation.

INPUT: <file_name>;

directs the command processor to read subsequent commands from the specified file.

HELP;

lists the currently implemented commands and their syntax.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER IPL-TR-033	2. GOVT ACCESSION NO. AD-A132 392	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Generating Three-Dimensional Surface Models of Solid Objects From Multiple Projections		5. TYPE OF REPORT & PERIOD COVERED Technical
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael Potmesil		8. CONTRACT OR GRANT NUMBER(s) N00014-82-K-0301
9. PERFORMING ORGANIZATION NAME AND ADDRESS Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, New York 12181		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research, 800 North Quincy Street, Arlington, VA 22217		12. REPORT DATE October 1982
		13. NUMBER OF PAGES 239
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

"The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notices heron."

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Graphics
Computer-aided design
3D modelling
Solid geometry
Image Processing
Pattern Recoginition

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A recurring problem in computer vision and related fields is that of generating computer models of physical objects. This thesis presents a method for constructing such models in the form of three-dimensional surface or volume descriptions. The surface models are composed of curved, topologically rectangular, parametric patches. The data required to define these patches are obtained from mulitple photographic images of the object illuminated by a rectangular pattern of lines. The projection of the pattern on the surface of the object (over)

traces curves which define the boundaries of the patches. The 3D description of the patches. The 3D description of the patches is reconstructed by photogrammetric techniques from two or more images of the projected pattern. A calibration stand, in which the object is placed, permits determination of the camera geometry directly from image data.

This method generated 3D surface descriptions of only those parts of the object that are illuminated by the projected pattern, and also are visible in at least two images. A complete model of the object is obtained by repeating this reconstruction process for various arbitrary orientations of the object until descriptions covering the entire surface have been obtained. Since each description is defined in its own 3D object space and 2D parameter space, a surface-matching procedure is developed to register spatially the surface descriptions in a common object space. This procedure searches for a 3D rigid transformation of the surface descriptions which minmizes their shape difference. Once the surface descriptions are in the same object space, they are also merged into a common parameter space. This match-and-merge process is iteratively repeated for pairs of surface descriptions until a completer model of the object is assembled.